'68' MICRO JOURNAL

$2.50 USA

Australia	A $ 4.00	New Zealand	NZ $ 4.00
Singapore	S $ 8.00	Hong Kong	H $20.00
Malaysia	M $ 8.00		

VOLUME I ISSUE 8 • Devoted to the 6800 User • October 1979

"Small Computers Doing Big Things."

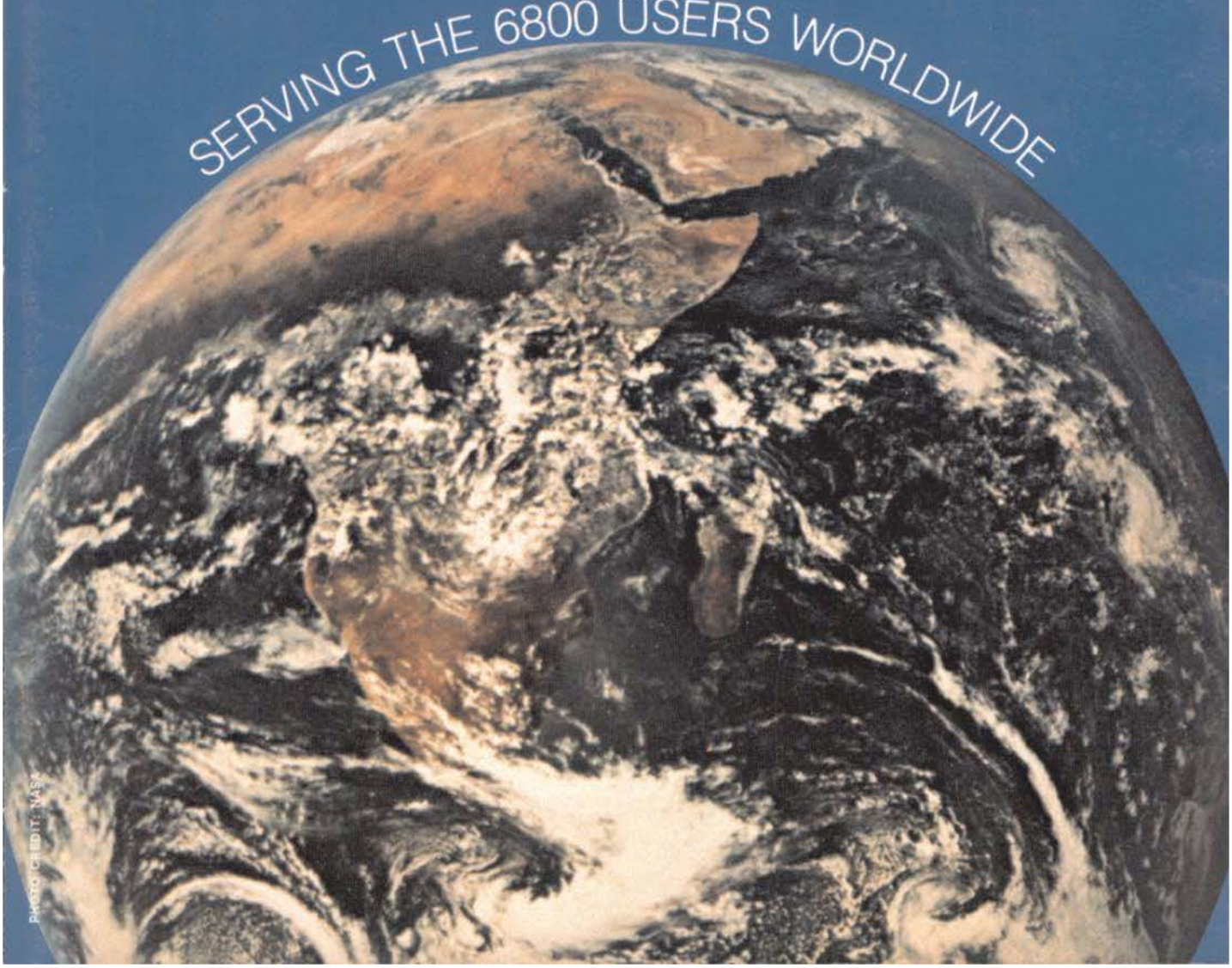


PHOTO CREDIT: NASA

SYSTEMS - SOLUTIONS

If you have a problem that can be solved by a computer—we have a systems solution.

- Two central processors with maximum RAM capacities of 56K and 384 K bytes
- Three types of disk drives with capacities of 175K, 1.2M and 16M bytes
- Two dot matrix printers with 80 and 132 line capacity
- A Selectric typewriter interface and a daisy wheel printer

Match these to your exact need, add one or more of our intelligent terminals and put together a system from one source with guaranteed compatibility in both software and hardware.

Southwest Technical Products systems give you unmatched power, speed and versatility. They are packaged in custom designed woodgrain finished cabinets. Factory service and support on the entire system and local service is available in many cities.

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Something New on the Horizon from Technical Systems Consultants

Extended BASIC for 6800 and 6809

Finally, a BASIC for serious business applications or scientific programming is available. All the features of our regular BASIC are supported—and more. Floating point calculations are carried out to an internal accuracy of 17 digits. Most math functions are accurate to 16 digits with a minimum accuracy of 13.5 digits. Integer variables have been included to allow fast execution of control loops and array indexing. Even with the double precision math package, this BASIC is still one of the fastest around.

The business programmer will appreciate the versatile PRINT-USING capabilities which include dollar and asterisk fill, trailing minus sign, imbedded commas, and scientific notation. New string functions have been added for string searching (INSTR) and for creating a string which is the date (DATE$). DPEEK and DPOKE are 16-bit peek and poke type functions. The SCALE command has been included to eliminate the round-off errors typically encountered in binary math packages. The INCH$ function allows single-character input from the terminal. Programmer control of control C breaks is also included.

Overall, the Extended BASIC is the most complete BASIC offered for micro users and is only available on FLEX™ disk. A system with at least 32K of user space is recommended. Specify 8" or 5" media (5" 6800 is FLEX™ 2.0) and either the 6800 or 6809 version when ordering.

AP68-12	**6800 Extended BASIC**	**$100**
SP09-6	**6809 Extended BASIC**	**$100**

BASIC Precompiler

This program allows the creation of BASIC programs without the use of line numbers or restrictive two-character variable names. Alphanumeric line and subroutine labels may be used, as well as variable names of any length. Comment lines are marked with non-alphanumerics for easy readability. The output of the precompiler is in the standard BASIC compiled form. This allows applications programs to be written, precompiled, and then distributed in a non-source form. The precompiler can only be used with one of Technical Systems Consultants' BASICs. Specify 8" or 5" (5" 6800 is FLEX™ 2.0) when ordering.

AP68-13	**Single Precision 6800 Precompiler**	**$40**
AP68-14	**Double Precision 6800 Precompiler**	**$50**
SP09-7	**Single Precision 6809 Precompiler**	**$40**
SP09-8	**Double Precision 6809 Precompiler**	**$50**

FLEX is a registered trademark of Technical Systems Consultants, Inc.

technical systems consultants, inc.
Box 2570, West Lafayette, IN 47906
(317) 463-2502

'68' MICRO JOURNAL

Portions of the text of '68' Micro Journal set using the following:
6800/2, DMAF1 and CT-82
Southwest Technical Products Corp.
219 W. Rhapsody
San Antonio, TX 78216

Editor, Word Processor and Sort/Merge
Technical Systems Consultants, Inc.
Box 2574
W. LaFayette, IN 47906
*MINIFLEX & FLEX REG.™ Technical Systems Consultants, Inc.

Selectric I/O
World Wide Electronics, Inc.
130 Northwestern Blvd.
Nashua, NH 03060

Publisher/Editor
Don Williams Sr.

Executive Editor
Larry E. Williams

Assistant Editor — Software
Mickey E. Ferguson

Assistant Editor — Hardware
Dennis Womack

Associate Editor — Southwest
Dr. Jack Bryant

Associate Editor — At Large
Dr. Chuck Adams

Associate Editor — Midwest
Howard Berenbon

Contributing Editors
Dr. Jeffrey E. Brownstein
Dale Puckett

Subscriptions and Office Manager
Joyce Williams

Typography and Color Separations
Williams Company, Inc.
Chattanooga, TN 37421

Contents

Send All Correspondence To:

'68' Micro Journal
3018 Hamill Rd.
PO Box 849
Hixson, Tennessee 37343

— Phone —
Office: 615-870-1993
Plant: 615-892-7544

'68' Micro Journal is published 12 times a year by '68' Micro Journal, 6131 Airways Blvd., Chattanooga, TN 37421. Second Class postage paid at Chattanooga, TN. Postmaster: Send Form 3579 to '68' Micro Journal, PO Box 849, Hixson, TN 37343.

1-Year $14.50 2 Years $26.00 3 Years $36.50

—ITEMS SUBMITTED FOR PUBLICATION—

(Letters to the Editor for Publication) All 'letters to the Editor' should be substantiated by facts. Opinions should be indicated as such. All letters must be signed. We are interested in receiving letters that will benefit or alert our readers. Praise as well as gripes is always good subject matter. Your name may be withheld upon request. If you have had a good experience with a 6800 vendor please put it in a letter. If the experience was bad put that in a letter also. Remember, if you tell us who they are then it is only fair that your name 'not' be withheld. This means that all letters published, of a critical nature, cannot have a name withheld. We will attempt to publish 'verbatim' letters that are composed using 'good taste.' We reserve the right to define (for '68' Micro) what constitutes 'good taste.'

(Articles and items submitted for publication) Please, always include your full name, address, and telephone number. Date and number all sheets. TYPE them if you can, poorly handwritten copy is sometimes the difference between go, no-go. All items should be on 8X11 inch, white paper. Most all art work will be reproduced photographically, this includes all listings, diagrams and other non-text material. All typewritten copy should be done with a NEW RIBBON. All hand drawn art should be black on white paper. Please no hand written code items over 50 bytes. Neatly typed copy will be directly reproduced. Column width should be 3¼ inches.

(Advertising) Any Classified: Maximum 20 words. All single letters and/or numbers will be considered one (1) word. No Commercial or Business Type Classified advertising. Classified ads will be published in our standard format. Classified ads $7.50 one time run, paid in advance.

Commercial and/or Business advertisers please write or phone for current rate sheet and publication lag time.

SEE GHOST AD PAGES 38 & 43

A.C. POWER CONTROL for ALL COMPUTERS or COMPLETE TURNKEY SYSTEMS

Interface TO the Real World with GIMIX Relay Driver Boards. Connects to any Computer through a 20 ma. current loop (up to 4 Boards - 128 Relays per port).

Interface FROM the Real World with GIMIX

- ★ OPTO BOARDS (up to 34 switch closures with one 8 bit Parallel I/O Port)
- ★ 16 BUTTON KEYPADS
- ★ 35 BUTTON ALPHANUMERIC KEYPADS

A Broad Range of 6800 Systems and Boards Compatible with Other SS50 Products such as SWTP and MSI

MAINFRAME: Includes chassis, power supply, switches, fan and mother board . . **$ 798.19**

16K SYSTEMS: Mainframe, plus 6800 CPU, 16K Static Ram and choice of I/O . . . **$1344.29**
Other packages available.

6800/6809 MOTHER BOARD

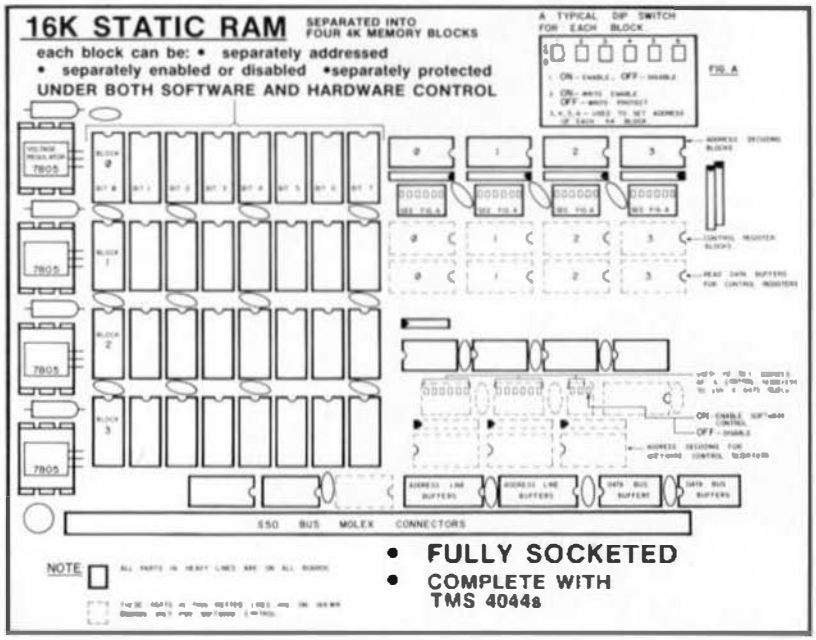
Hardware reconfigurable to give you the utmost versatility for use with various 6800 and 6809, SS50 and SS50C bus configurations. Gold plated pins to insure long lasting electrical contact for protection against corrosion. Fifteen 50 pin slots plus eight DIP-switch addressable 30 pin I/O slots configurable to 4, 8 or 16 decoded addresses. The fully buffered I/O block is addressable by DIP-switch to any 32, 64 or 128 byte boundary and can also be disabled. An on board baud rate generator provides 9 standard baud rates from 75 to 9600. A 14 pin DIP-header allows easy selection of up to five baud rates. The five baud rate lines on the 50 pin bus can be easily disconnected from the 30 pin bus for use as user defined lines (extended address lines etc.) An optional slow I/O circuit, for the 6809 CPU, allows stretching the clock ½ cycle whenever an I/O device is accessed. (This allows, for example, using 1Mhz I/O cards with a 2 Mhz 6809 CPU.) UD1 and UD2 of the 50 pin bus can be strapped to UD3 and UD4 of the 30 pin bus. A fully shielded, (.093" thick), double sided P.C. board with noise reducing ground lines on the bottom side that separate all data, address, and signal lines, and a full ground plane on the top side. Schmitt trigger buffering of all address, data, and control signals. A 14 position clamping terminal block for all power and other external connections eliminates soldering, crimping or forming of wires.

TI TMS 4044's — 10% SUPPLY
(Not an "equivalent", but the real thing!)

450 ns	$5.90 each	250 ns	$6.90 each
EXPROM BOARD			**$ 98.34**
4K PPD PROM BOARD, Burner and Duplicator			**198.35**
2708's FACTORY PRIME 450 ns		each	**7.90**
64 or 32 x 16 VIDEO BOARD			**198.71**
80 x 24 SUPER VIDEO BOARD with user programmable RAM character generator			**458.76**
Serial I/O's	1 Port $ 88.41	4 Port	**198.43**
Parallel I/O's	2 Port $ 88.42	8 Port	**198.45**

Add $5. handling charge on orders under $200.

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055
Quality Electronic products since 1975.

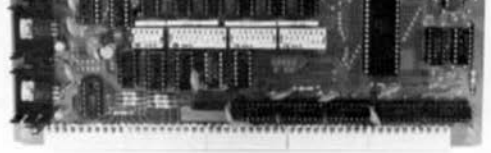


and its also SOFTWARE CONTROLLABLE $368.16

Under software you can program each 4K blocks address and you can enable/disable or write protect each 4K block. The CPU can read the status of the software control registers for each 4K. This allows expansion of memory way beyond 65K; or allows multi-tasking with just one 16K board; all with very little software overhead. *(We have calculated that you can put 13,000 of these boards in a system — that's 208 megabytes, but we do not intend to make either a bus or power supply that size.)* All registers included — no phantom lines needed.

- ★ Facilitates multiprogramming, time-sharing, and sofware development.
- ★ All Gimix memory boards are assembled, burnt-in for 2 weeks, and tested at 2Mhz.

AS ABOVE, BUT WITHOUT SOFTWARE CONTROL AND UNSOCKETED $298.13



GIMIX 6800 CPU BOARD

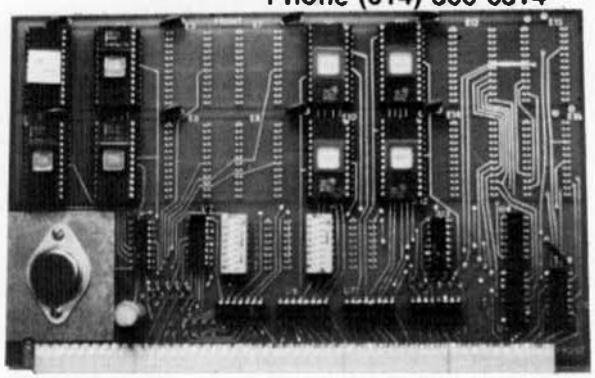
- ★ 6800 MPU ★ 4K EPROM
- ★ 3 Programmable timers
- ★ 128 byte RAM
- ✓ DIP-switch addressing and enabling or disabling for software use versatility.

This board features:
- ✓ Crystal controlled 6800 MPU using a 6875 clock generator
- ✓ 14411 bit-rate generator with its own crystal that provides baud rates from 110 to 9600.
- ✓ 128 Byte 6810 SCRATCHPAD RAM, which can be DIP-switch addressed to any 128 byte boundary, or disabled.
- ✓ Gold Bus Connectors

Buffered for reliability and data integrity All address output, data input and output, halt input, clock, reset, control and baud rate output lines are buffered. Manual reset input line is buffered and debounced. IRQ and NMI lines are direct in (6.8K pullup) DMA capability through cycle stealing or halt.

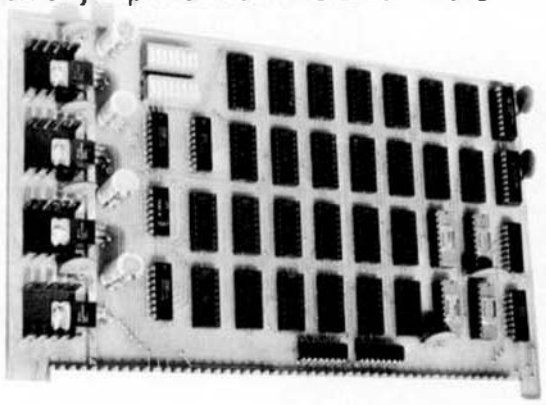
A 4K ROM SECTION THAT HAS:
Sockets for 4 2708 EPROMS which can be DIP-switch addressed to any 4K boundary, or disabled. Dual-address switch lets one PROM respond to both E000 and FC00, for MIKBUG compatability. Split-address strapping places PROMS at E000, E400, E800, FC00 if desired.

Plus Three Programmable Timers (optional)
Eliminate the need for software timing loops. A 6840 software programmable timer provides 3 independent 16-bit counters which may be used to cause interrupts and/or generate output signals. They can be used individually or in combination. They may be programmed for one-shot timing or for regular intervals. Applications for the timers include frequency measurements, event counting, interval measuring, and similar tasks. They can be used for square wave generation, gated delay signals, single pulses of controlled duration, and pulse width modulation. DIP-switch addressable to any 8 byte boundary. Additional DIP-switch positions control IRQ or NMI interrupt choices, as well as enabling or disabling the timers.



DSD P/R-32K $27.00

32K or 16K EPROM & RAM memory card. 2716 2K x 8 or 2758 1K x 8 5V only EPROMS. TMS 4016 2K x 8 or MK 4118 1K x 8 5V RAMS. Up to 4 independent addressed 8K blocks. Dip switch or jumper selected. Size 9" x 5½"



DSD 2114-16K $27.00

Full Static 16K Ram memory card designed to use the 2114 or TMS 4045 1024 x 4 Static Ram. The card has two independent addressed 8K memory blocks. Card size 9" x 5½". Power requirements 7-8V unreg. @ 3.5A.

DSD U P 8255M $14.00

Universal parallel interface card with wire wrap area using INTEL'S 8255 parallel peripheral interface chip. 24 programmable I/O lines. (Three 8 bit Ports or Two 8 bit Ports with handshaking) Card size 5½" x 5" Standard SS-50 30 pin I/O BUS. 5V only.

Cards are bare with data and edge connector. Ohio residents add 4½% sales tax.

6847 Color Graphic card in design

HELP

I am building a small data logger using a 6800 microprocessor. I have access to a Heathkit H89 system (two Z80's with a floppy disk system) and would like to develop and store my 6800 software on this system. Do you know where I can acquire 6800 cross assembler software to run on the Heathkit Z80 system. Please forward any info to James A Coleman, 117 Frost Lane, Newton, Pa. 18940

Ed's Notes:

James, I know of no cross assembler to the Heath equipment. I honestly don't think there will be any. It seems a shame to try to run perfectly good code on a system that is not, to say the least, what it should be. Two Z80's are not the answer. With Heath equipment I don't know.

Better look for another way James.

DMW

1-Year $14.50 2 Years $26.00 3 Years $36.50

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card #_____ Exp. Date_____

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: $_____

Name_____

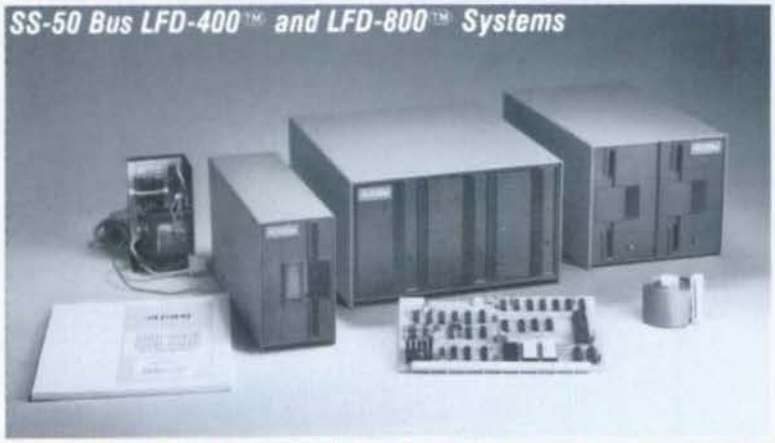
Street_____

City_____ State_____ Zip_____

My Computer Is:_____

68 MICRO JOURNAL
3018 Hamill Road
HIXSON, TN 37343

Welcome to Percom's Wide World



Each LFD mini-disk storage system includes:

- drives with integral power supplies in an enamel-finished enclosure
- a controller/interface with ROM operating system plus extra ROM capacity
- an interconnecting cable
- a comprehensive 80-page users manual

Low-Cost Mini-Disk Storage in the Size You Want.

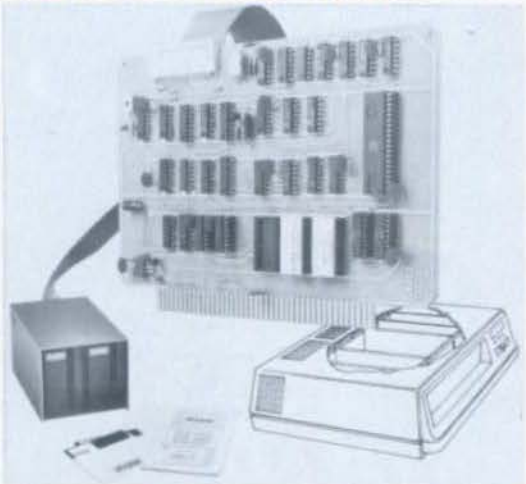
Percom LFD mini-disk drive systems are supplied complete and ready to plug in the moment they arrive. You don't even have to buy extra memory. Moreover, software support ranges from assembly language program development aids to high-speed disk operating systems and business application programs.

The LFD-400™ and -400EX™ systems and the LFD-800™ and -800EX™ systems are available in 1-, 2- and 3-drive configurations. The -400, -400EX drives store 102K bytes of formatted data on 40-track disks, and data may be stored on either surface of a disk. The -800, -800EX drives store 200K bytes of formatted data on 77-track disks.

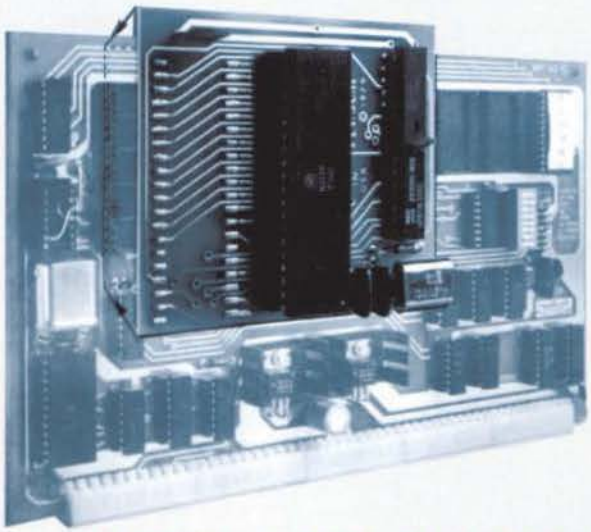
The LFD-1000™ systems (not pictured) have dual-drive units which store 800K bytes on-line. The LFD-1000™ controller accommodates two drive systems so that a user may have as much as 1.6M bytes on-line.

Mini-disk storage system prices:

MODEL	1-DRIVE SYSTEM	2-DRIVE SYSTEM	3-DRIVE SYSTEM
For the SS-50 Bus:			
LFD-400™	$ 599.95	$ 999.95	$1399.95
LFD-800™	895.95	1549.95	2195.95
For the EXORciser® Bus:			
LFD-400EX™	$ 649.95	$1049.95	$1449.95
LFD-800EX™	945.95	1599.95	2245.95
LFD-1000™	(dual) $2495.00	(quad) 4950.00	—



EXORciser® Bus LFD-400EX,™ -800EX™ Systems



Upgrade to 6809 Computing Power. Only $69.95

Although designed with the SWTP 6800 owner in mind, this upgrade adapter may also be used with most other 6800 and 6802 MPUs. The adapter is supplied assembled and tested, and includes the 6809 IC, a crystal, other essential components and user instructions. Restore your original system by merely unplu ing the adapter and a wire-jumpered DIP header, and re-inserting the original components. Also available for your upgraded system is PSYMON™ (Percom SYstem MONitor), the operating system for the Percom 6809 single-board computer. PSYMON™ on 2716 ROM costs only $69.95. On diskette (source and object files), only $29.95.

Data Terminal & Two-Cassette Interface — the CIS-30+

- Interface to data terminal and two cassette recorders with a unit only 1/10 the size of SWTP's AC-30.
- Select 30, 60 or 120 bytes per second cassette interfacing; 300, 600 or 1200 baud data terminal interfacing.
- Optional mod kits make CIS-30+ work with any microcomputer. (For MITS 680b, ask for Tech Memo TM-CIS-30+-09.)
- KC Standard/Bi-Phase-M (double frequency) cassette data encoding. Dependable self-clocking operation.
- Ordinary functions may be accomplished with 6800 Mikbug® monitor.

Prices: Kit, $79.95; Assembled, $99.95. Prices include a comprehensive instruction manual. Also available: Test Cassette, Remote Control Kit (for program control of recorders), IC Socket Kit, MITS 680b mod documentation and Universal Adapter Kit (converts CIS-30+ for use with any computer).

of 6800 Microcomputing.

6800/6809 SOFTWARE

System Software

6800 Symbolic Assembler — Specify assembly options at time of assembly with this symbolic assembler. Source listing on diskette $29.95

Super BASIC — a 12K extended random access disk BASIC for the 6800 and 6809. Supports 44 commands and 31 functions. Interprets programs written in both SWTP 8K BASIC (versions 2.0, 2.2 & 2.3) and Super BASIC. Features: 9-digit BCD arithmetic. Print Using and Ungot commands, and much more. Price $49.95

TOUCHUP™ — Modifies TSC's Text Editor and Text Processor for Percom mini-disk drive operation. Supplied on diskette complete with source listing $17.95

Operating Systems

INDEX™ — This easy-to-use disk-operating file management system for 6800 microcomputers is fast. I/O devices are serviced by interrupt request. INDEX™ accesses peripherals the same as disk files — new devices may be added without changing the operating system. Other features: unlimited number of DOS commands may be added • over 60 system entry points • display only those files at or above user-specified file activity level • versions available for SWTP MF-68, Smoke's BFD-68 and Motorola's EXORciser. Price $99.95

MINIDOS-PLUSX™ — An extension of the original MINIDOS™ for LFD-400™ mini-disk systems, MINIDOS-PLUSX™ manipulates files by six-character names. Supports up to 31 files. Resident commands include Initialize, Save, Allocate, Load, Files (directory list), Rename and Delete. Supplied on 2708 ROM with a minidiskette that includes transient utilities such as Copy, Backup, Create, Pack and Print Directory. Price ..... $34.95.

PSYMON™ — Percom SYstem MONitor for the Percom single-board/SS-50-bus-compatible 6809 computer accommodates user's application programs with any mix of peripherals **without** modifying programs. PSYMON™ also features character echoing to devices other than the communicating device, sophisticated register and memory dump routines and more. Price (on 2716 ROM) $69.95

WINDEX™ — Described in detail elsewhere on this page.

Business Programs

General Ledger — For 6800/6809 computers using Percom LFD mini-disk storage systems. Requires little or no knowledge of bookkeeping because the operator is prompted with non-technical questions during data entry. General Ledger updates account balances immediately — in real time, and will print financial statements immediately after journal entries. User selects and assigns own account numbers; tailors financial statements to firm's particular needs. Provides audit trail. Runs under Percom Super BASIC. Requires 24K bytes of RAM. Supplied on minidiskette with a comprehensive users manual. Price ..... $199.95.

FINDER™ — This general purpose data base manager is written in Percom Super BASIC. Works with 6800/6809 computers using Percom LFD-400™ mini-disk drive storage systems. FINDER™ allows user to define and access records using his own terminology — customize file structures to specific needs. Basic commands are New, Change, Delete, Find and Pack. Add up to three user-defined commands. FINDER plus Super BASIC require 24K bytes of RAM. Supplied on minidiskette with a users manual. Price $99.95

Mailing List Processor — Powerful search, sort, create and update capability plus ability to store 700 addresses per minidiskette make this list processor efficient and easy to use. Runs under Percom Super BASIC. Requires 24K bytes of RAM. Supplied on minidiskette with a users manual. Price $99.95.

From the Software Works

Development and debugging programs for 6800 µCs on diskette:

Disassembler/Source Generator	$30.95
Reloc'tng Disas'mblr/Segmented Text Gen	$40.95
Disassembler/Trace	$25.95
Support Relocator Program	$25.95
Relocating Assembler/Linking Loader	$55.95
SmithBUG** (2716 EPROM)	$70.00

½-Price Special on Hemenway Software!

CP/68‡ disk operating system	$ 49.97
STRUBAL+‡ compiler	$124.97
EDIT68 text editor	$ 19.97
MACRO-Relocating Assembler	$ 39.97
Linkage Editor (LNKEDT68)	$ 24.97
Cross Reference utility	$ 14.97

™ Trademark of Percom Data Company, Inc.
* Trademark of Motorola Corporation
‡ Trademark of Hemenway Associates Company
** SmithBUG is a trademark of the Software Works Company

And 'looking into' is just what you do with the Electric Window™ as you peer right into memory space where characters are being input and manipulated. Display is memory-resident, programmable and generates up to 24 80-character lines.

Other features include:

- standard character generator plus provision for optional special character generator
- dual intensity, high-lighting alphanumeric display
- scrolling by a programmable register • programmable display positioning
- programmable interlaced or non-interlaced scan
- descenders on lower case letters • users manual with application instructions and listing of WINDEX™ driver.

The Electric Window.™ Worth Looking Into. $249.95

WINDEX™ is a fast video display driver program for the Electric Window™. WINDEX™ also features: program and keyboard control of character generators • displayable control characters — under program control • automatic scrolling • a driver routine for the parallel input keyboard feature of the Percom 6809 Single-Board Computer, the SBC/9™ • auto-linking to PSYMON™, the ROM operating system for the SBC/9™ • Prices: ROM version: $39.95; LFD-400™ compatible diskette (source and object files): $29.95.

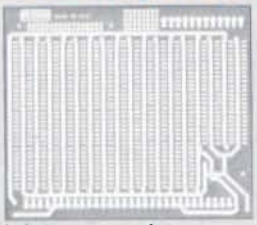
Now Available! the SBC/9™ MPU/Control Computer

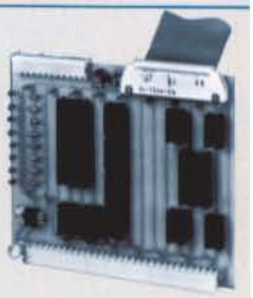
(Single-Board-Computer/6809) — stands alone as a control computer, but also compatible with the SS-50 bus for use as an MPU card. Includes PSYMON™ (Percom SYstem MONitor) in a 1K ROM and provides for additional 1K of ROM. Also includes 1K of RAM. Features: Super Port — provision for multi-address, 8-bit bidirectional data lines • an intelligent data bus for multi-level data bus decoding • an on-board 110-baud to 19.2 kbaud clock generator • extended address capability — to 16 megabytes — without disabling baud clock or adding hardware. And much more. Supplied with PSYMON™ and comprehensive users manual. Price ..... $199.95.

See full page ad elsewhere in this magazine for all of the SBC/9™ features.

Full Feature Prototyping PC Boards

All of the features needed for rapid, straightforward circuit prototyping. Use 14-, 16-, 24- and 40-pin DIP sockets • SS-50 bus card accommodates 34- and 50-pin ribbon connectors on top edge, 10-pin Molex connector on side edge • I/O card accommodates 34-pin ribbon connector and 12-pin Molex on top edge





I/O Bus Card: $14.95

SS-50 Bus Card: $24.95

- I/O card is 1-¼ inches higher than SWTP I/O card • interdigitated power conductors • contacts for power regulators and distributed capacitance bypassing • use wire wrap, wiring pencil or solder wiring • tin-lead plating over 2-oz copper conductors wets quickly, solders easily • FR4-G10 epoxy-glass substrate.

To place an order or request additional literature call toll-free 1-800-527-1592. For technical information call (214) 272-3421. Orders may be paid by check, money order, COD or charged to a VISA or Master Charge account. Texas residents must add 5% sales tax.

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

DIGITAL RESEARCH: COMPUTERS S50 MEMORY BOARD 16K

A 68 Micro Journal™ Lab Review

A new company, that is not really too new, has recently announced a new S50 16K static memory card. It may be ordered assembled, in sockets, tested and guaranteed, $295.00. Blank board $33.00; complete socket set $12.00; support IC's and caps $19.95.

Many of you will remember S D Sales. They made (and still do) many kits for not only computers, but many other areas of electronic endeavor. Also they were one of the first to offer reasonably priced parts (IC's, transistors, etc.) to the hobbyist. Well it seems that S D Sales has new owners and Digital Research: Computers is the effort of some of those who originally started S D Sales.

Features

Addressable on 16K boundaries by jumper selection or 8 pin dip switch (dip switch not included). Uses 2114 static RAM memory chips. The printed circuit board is double sided, solder mask with silk screen layout. The 50 pin connector strip contains gold plated fingers. Typical power drain less than 2 amps.

One nice feature we 'discovered' on the card is that the four (4) 7805 5v regulators are installed at the very top of the card, 4 across, all with heavy (for a 7805) heatsinks. This is an example of good engineering, especially where heat or thermal reaction might be a problem, in some mainframes. Due to the top location, very little heat generated by this board, convects up over it's or any other boards components. We have known many instances where this type of engineering would have made a vast difference. The memory board runs very cool, due to this design feature.

Construction

The Digital Research: Computers S50 static memory board we tested came factory assembled. However, as is our policy, if a kit is available, we require the construction manual, for the item. This just as it would be sent one of our readers, that placed an order for the memory board. The construction manual is 7 pages, one of which is a clearly drawn circuit diagram. The assembly instructions are more than adequate, even for the less experienced builder. Addressing options are explained and a short 'theory of operation' section is included.

Conclusion

We have experienced no problems with our test memory board. It performs well and is engineered for efficiency. All the components are of good quality and the board contains large ground plane area for for low noise. The bypassing is more than sufficient with both discs and tantalum capacitors. All address and data lines are buffered. Board access time is rated under 500 ns. Additional information can be secured from:

Digital Research: Computers
PO Box 401565
Garland, TX 75040
(214) 494-1505

A 68 Micro Journal™ Lab rating: AAA

Rating Scale:
AAA - Excellent
AA - Good
A - Fair (could be better but works)
P - Poor (may not always work properly)
X - Not recommended for children
(or anything else!)

August 20, 1979

HELLO,
I'm sending along several more "BASIC PROGRAMMING QUICKIES" written in SWTPC 8K Basic, V2.3
Of particular interest is the subroutine which computes the JULIAN DATE.

```
9600  REM MATRIX MULTIPLICATION SUBROUTINE
9610 REM
9620 REM INPUTS
9630 REM MATRIX A, M ROWS BY N COLUMNS
9640 REM MATRIX B, N ROWS BY P COLUMNS
9650 REM
9660 REM OUTPUT
9670 REM MATRIX C, M ROWS BY P COLUMNS
9680 REM
9690 FOR I=1 TO M:FOR J=1 TO P:S=0
9700 FOR K=1 TO N:S=S+A(I,K)*B(K,J):NEXT K
9710 C(I,J)= S:NEXT J:NEXT I:RETURN


8000 REM DAYS BETWEEN DATES SUBROUTINE
8010 REM
8020 REM INPUTS
8030 REM M= MONTH, D= DAY, Y= YEAR
8040 REM
8050 REM OUTPUT
8060 REM F= FACTOR
8070 REM COMPUTE FACTORS FOR TWO DATES
     IN QUESTION
8080 REM DAYS BETWEEN DATES=FACTOR(DATE 1)
     -FACTOR(DATE 2)
8090 REM
8100 IF M> 2 GOTO 8120
8110 F=365*Y+D+31*(M-1)+INT((Y-1)/4)
     -INT(.75*(INT(((Y-1)/100)+1))):RETURN
8120  F=365*Y+D+31*(M-1)-INT(.4*M+2.3)+INT
     (Y/4)-INT(.75*(INT(Y/100)+1)):RETURN


9600 REM JULIAN DATE AT 0 HOURS U.T.
     SUBROUTINE
9610 REM
9620 REM INPUTS
9630 REM M= MONTH, D= DAY, Y= YEAR
9640 REM
9650 REM OUTPUT
9660 REM J= JULIAN DATE AT 0 HOURS U.T.
9665 REM
9670 X= INT((M-14)/12)+1:IF M=2
     THEN X= INT((M-14)/12)
9680 A= D-32075+INT
     (1461*(Y+4800+X)/4)
9690 B= INT(367*(M-2-X*12)/12)
9700 C= -INT(3*((Y+4900+X)/100)/4)
9710 J= A+B+C-.5:RETURN
```

Sincerely,
DAVID EAGLE
3330 S. Garland Way
Lakewood, Co. 80227

SOFTWARE UPDATE

It seems that many readers are interested in coming software, for both the 6800 and 6809.

Many readers call us to see if we know when certain utilities or major programs will be released. We try to get this type of information in the form of 'press releases' which we in turn reproduce in 68 Micro Journal. We encourage all vendors of 6800/09 products to keep us informed. Even if they do not advertise in 68 Micro Journal, we want to tell you what is available and at what price and when. This is our job; keeping you informed with as timely information as possible. To you vandors who are swamped (we know what that is) if you cannot get out a 'release' please than a short pencilled note letting me know what the new offerings are, their price and anticipated availability dates. This way maybe it will cut down some on your inquiry calls and let our readers, your customers, know.

We are fortunate in the 6800/09 community to have good hardware and software vendors. There is very little 'junk' being offered for sale to the 6800/09 group. There is more good software and hardware available today for us than any other user group. Our big problem is finding out what, when and the price, mostly in that order. By working together we all benefit.

From Technical System Consultants comes the following. A package of 16-18 utilities for 09 FLEX, should be available in October, probably by the time you receive this issue. 6800-6809 pre-compilers for TSC BASIC in October also. For the 6809 4 user, 128K SWTPC systems the following in Octobar also; Sort-Merge, an improved and upgraded Processor, a new Text Editor and Assembler and sevaral compilers. In addition an Oparation System is in the offering.

Also by the time you receive this the extended precision or 'business' version of TSC BASIC should be available. We were informed also that the 5" version of 09 FLEX delivery started in August. So to all of you who called, there it is.

Smoke Signal Broadcasting has begun shipping the new version of SSB Dos. It has many additions and improvements over version 4. For you GIMIX GMXBUG ver. 3 users, you will be glad to know that the copy furnished us does not conflict with this monitor.

Computerware is also now shipping their new 'random' BASIC. Available is a selection of commercial packages - business applications and utilities. We are informed that many of the Computerware offerings are available for FLEX users also.

The naw lineup of Ed Smith Software Works for the 09 should be being shipped by now.

Most all the software that was available for the 6800 is now or will shortly (will try to let you know) available for you who are into the 6809.

In this issue is a new 6809 monitor 'PSYMON' developed by PERCOM. For those wishing to 'roll their own' this is a good start. It works with PRECOM's new 6809 adaptor and can be adapted to most any 6800 system. It has been released to 'unrestricted Royalty-Free License' status, by Harold Mauch, President of PERCOM. Dr. Chuck Adams, our Associate Editor, is doing an adaptation for other 6809 systems. We will publish it as soon as it is available.

LUCIDATA wrote that they now have PASCAL, with some new 'goodies' included for the 6809. Also those who now have this software may upgrade for a nominal amount.

I try to keep up with what is happening. I appreciate your confidence in 68 Micro Journal. I know that sometimes you vendors get tired of me 'bugging' you for information. Yet, by a survey we conducted in June, I found out that information about 'new products' for the 6800 and 6809 is right up at the top of the things that our readers (and your customers) want to see in 68 Micro Journal. This is thousands of actual users, not 'tire kickers' or 'curosity seekers'. We can, if necessary, get your information out within 15 days of our next publication. In other computer magazines I know for a fact that it takes 3 to 10 times this long. It is not easy nor is it normally expected. I just want our readers to 'stay on top' of all the new software and hardware available. Especially important is ACTUAL shipping dates. I receive many complaints about advertising before a product is near completion. If I know the product is not ready, I try to let the readers know. That way is best for all of us, and possibly inhibits some customer hard feelings. Again I repeat; new product insertions and product reviews are among the top, in reader interest.

It may not be this way always, I rather think not, but for now our market is not 'glutted' with a lot of vendors, all attempting to sell the same fuctional product, in the same marketplace. Our choice of products is varied, each serving a needed and useful purpose. It might be said that we are still in a 'sellers market', Someday it will be different, the base that we build on now will support our continued operation, to the degree that we develop and retain buyer confidence while improving and expanding our product. If any 6800/09 vendor fails we all lose something. As each prospers we all prosper. None of us is going to hack it alone. Sorta seems like a worldwide club to us here. Standing somewhat in the middle, as we do, I see exciting things just around the corner of time. If we remain a cohesive group we will all benefit, to the degree that we contribute. We intend to be around a long time, I hope you will also, user and seller alike.

PERCOM
211 N. Kirby
Garland, TX 75042

PYSOM—6809 MONITOR

INTRODUCTION

PSYMON, the Percom SYstem MONitor for the 6809, is a simple 1K operating system designed for the Motorola 6809 microprocessor. While it provides commands for program loading and saving, memory and register examine/change, and breakpoint management, the true power of PSYMON is in its structure and extensibility.

PSYMON was designed to be as easy as possible to interface to regardless of the hardware environment. It may be highly customized and extended due to its unique "look-ahead" and device independent I/O structure. This adaptability was the result of the use of structured techniques in the design and programming of PSYMON. The members of the design team were Harold Mauch, Mike Foreman, Byron Seastrunk, Cliff Rushing, and Jim Stutsman. All of these team members have extensive experience with a variety of monitors for the MC6800 from which to draw.

DESCRIPTION OF COMMANDS

When PSYMON first receives control (usually through the power-on vector of the 6809 processor) it initializes its RAM areas, configures its console, and looks ahead for a second PROM (more about this later). At this time PSYMON will prompt with 'CMD?' and wait for the input of a legal command. All commands consist of a single letter. Some require parameters in the form of address or data. Whenever hexadecimal data is input to PSYMON, it is accepted according to a simple scheme. First, any non-hex character (other than 0-9 or A-F) terminates the hex entry. Certain "terminator" characters may have special meaning depending on the command. Second, leading zeroes are assumed on all entries shorter than the required size. For example, entry of FE as a parameter for an address would be interpreted as 00FE. Finally, if more digits are entered than are expected, only the last ones entered are used. For example, if 12345 is entered when a single byte is expected, the value used will be 45.

Command Set Summary

```
M <ADDRESS>          - MEMORY EXAMINE/CHANGE
G <ADDRESS>          - GO TO ADDRESS
R <REGISTER>         - REGISTER EXAMINE/CHANGE
L                    - LOAD PROGRAM (FROM TAPE)
S <START> <END>      - SAVE PROGRAM (TO TAPE)
B <ADDRESS>          - SET/LIST BREAKPOINTS
U <ADDRESS>          - UNSET BREAKPOINTS
Z                    - JUMP TO ADDRESS C000 (HEX)
```

M <address> - Memory examine and change

The command waits for an address to be entered. If a valid hex address is NOT entered, the LAST address examined is used (initially 0). This feature minimizes user frustration when inadvertently terminating a Memory Examine/Change sequence. It is also useful if you wish to repeatedly examine the same address (such as an I/O port).

First the address is displayed, followed by its contents in hex. The contents may be changed by entering a new value followed by a terminating character. If a new value is entered it is written into memory and verified. If the data did not store as expected, a '?' is displayed. Whether or not data was changed, the terminating character of the user entry is then examined. If the terminating character is '^', the address and content of the memory byte PRECEDING the one just examined will be displayed. The command then executes as previously described. If the terminating character is a CARRIAGE RETURN, the Memory Examine/Change is ended and control returns to the command prompt. Any OTHER terminating character will cause the address and content of the memory byte FOLLOWING the one just examined to be displayed and the examine/change process continues as described.

Examples:

```
M <TERM>            Displays last memory byte examined
                    (initially 0000)
M 1234<TERM>        Displays memory byte $1234
1234 F8 <SPACE>     SPACE causes display of NEXT byte
1235 F9 3F<SPACE>   F9 changed to 3F, display NEXT byte
1236 FA ^           No change, display PRECEDING byte
1235 3F <CR>        Carriage Return ends Examine/Change
CMD?
```

R <register> - Register examine and change

The command waits for the entry of a register name from the following list:

A - Accumulator A
B - Accumulator B
C - Condition code register
D - Direct page register
X - Index register X
Y - Index register Y
U - User stack pointer
P - Program counter

If no valid register name is entered, all registers are dumped and the command terminates. For a valid entry the contents of the register is displayed and the command waits for a replacement value to be entered. If a new value is entered it replaces the old value. In either case the command terminates and returns to the command prompt.

G <address> - Go to address

If a valid address is entered, it is placed in the Program Counter position on PSYMON's stack. If NO valid address is entered, the value already in the Program Counter position on the stack is used. All of the 6809 registers are loaded from PSYMON's stack (with an RTI instruction) and execution begins at the location pointed to by the program counter. Warning - the first thing user programs must do on receiving control is to establish a system stack (an LDS instruction). The stack space allocated for PSYMON is too limited for many applications. Failure to establish a new stack will result in the destruction of initial register settings.

L - Load a program from cassette

This command starts the cassette by raising the ACIA RTS (Reader Control) line. The tape is then scanned for records in the Motorola S1-S9 format. The load may be terminated in three ways:

1. Reception of an S9 record.
2. Detection of an invalid checksum.
3. Reception of a non-hex character in an S1 record.

In the case of 2 and 3 a '?' will be printed on the console. Note that tape I/O may be tailored to use other devices and techniques. This will be discussed later.

S <start> <end> - Save a program on cassette

The save command waits for user input of the starting and ending addresses of the memory to be saved on cassette. If only one address is entered, only the data at that address is saved. If NO address is entered, no data is saved and the actual save portion of the command is bypassed. Memory data is output to cassette in the standard Motorola S1 format. After all data has been saved the command terminating character entered by the user from the console is analyzed. If the terminating character is a CARRIAGE RETURN an S9 record is output to cassette. Any other terminator will suppress the S9 record. Finally control returns to the command prompt.

Examples:

S 100 3FF	Save memory from address $0100 through $03FF (no CR so no S9 record)
S 1000	Save byte from address $1000
S 500 7FF<CR>	The CR creates an S9 record after the data is saved
S <CR>	Output S9 record (no data)

B <address> - Set/list breakpoints

The command waits for entry of an address. If one is entered, and there is space in the breakpoint table (10

breakpoints maximum), the breakpoint is set and entered in the breakpoint table. In all cases all currently active breakpoints are listed. Warning - DO NOT breakpoint a location which already has a breakpoint. This condition will not be detected and will probably result in error.

U <address> - Unset a breakpoint

This command waits for input of a breakpoint address. If an address is entered the breakpoint table is searched for a match. When found, the breakpoint is removed. If the breakpoint cannot be found no action is taken. If no address is entered ALL active breakpoints are removed. Note - if a breakpoint is encountered during program execution, the breakpoint is automatically removed.

Z - Call PROM routine

This command, a relic from 6800 systems, is provided for user convenience. When entered, it performs a JSR to memory location $C000. Since PSYMON is designed to seek the highest level of existent operating system, this command will only be useful is the simplest systems.

PSYMON OPTIONS

PSYMON offers a rich variety of options which allow it to be tailored for nearly any configuration. This is done using the unique "look-ahead" feature. At power-up or reset, after initializing RAM and configuring the system console I/O device, PSYMON checks memory location F800. If a 7E (JMP instruction) is found PSYMON does a JSR to F800. This allows a user-written routine to alter any or all of the pointers used by PSYMON. To continue using revised RAM information the user routine need only do RTS (return from subroutine). Optionally the user routine may retain control and use PSYMON only for its subroutines.

All I/O in PSYMON uses a data structure known as a DEVICE CONTROL BLOCK (DCB). The DCB allows PSYMON to be relatively I/O device independent by leaving as much of the detail of the actual I/O as possible to the specific I/O device driver. The DCB is simply a table of parameters located somewhere in memory which among other things contains the address of the device driver routine. The Input/Output characteristics of the system may be subtly or radically altered by changing the contents of the DCB or by directing I/O through a different DCB. For example, data normally transmitted to the console terminal may be easily redirected to the printer or a disk. Likewise, a program may be loaded from a modem or disk instead of cassette tape by modifying the tape input DCB or by redirecting the input through another DCB.

The DCB is organized as follows:

Field	Offset	Usage
DCBLNK	0	Forward link in DCB chain (0 if last)

DCBDID	2	ASCII code for device identification
DCBDVR	4	Device driver address
DCBIOA	6	Device I/O address (meaningful to driver)
DCBERR	8	Error status code
DCBEXT	9	Number of extension bytes in DCB
DCBAPP	10	Optional appendage depending on driver

PSYMON itself has a single DCB which is used for all console functions. This DCB is initialized for I/O through an ACIA interface but may be altered since both the DCB and the pointers to the DCB are maintained in RAM. All keyboard input to PSYMON uses the DCB whose address is in CIDCB. Thus by changing this address, the input device alone may be changed. Echo of input characters is through the DCB pointed to by CEDCB. The input character echo is suppressed by setting CEDCB to zero. Output to the console device is through the DCB addressed by CODCB. All tape I/O uses the DCB pointed to by TPDCB. These pointers all initially point to CONDCB, PSYMON's console DCB. Any or all of the pointers may be changed by a user routine.

All of the hardware interrupts are vectored through addresses in PSYMON's RAM. SWI3V, SWI2V, and SWIV handle the various types of software interrupts. FIRQV is used for the "fast" interrupt while IRQV and NMIV are used for maskable and non-maskable interrupts respectively. A special vector, RESTRT, is provided for re-entry into PSYMON. This permits the normally unmodifiable RESET vector to be redirected. Initially SWI2V, SWI3V, IRQV, and NMIV are set to perform a register dump and return to the PSYMON command prompt. FIRQV initially points to an RTI (return from interrupt) instruction. SWIV points to PSYMON's breakpoint routine.

PSYMON's repetoire of commands is easily changed or enhanced. The pointer USRTBL in PSYMON's RAM contains the address of an alternate command table. It is initialized to zero, indicating no alternate table exists. This table, if used, must be constructed according to certain conventions. The first byte must be a 1, the length of a command in bytes. Each entry consists of a single ASCII character (the command) followed by the two-byte address of the routine which performs the command function. The end of the table is signified by a byte with bit 7 on (typically FF). Since the user table, if present, is always searched first, any or all of PSYMON's commands may be redefined by the user.

Command routines should preserve the U and S registers and should exit via an RTS (return from subroutine). Approximately 38 bytes of stack are available via the S register. If a larger stack is required, the user routine must provide for it.

PSYMON I/O

As previously mentioned, all I/O within PSYMON is handled using a Device Control Block (DCB). To perform I/O using a DCB it is first necessary to construct the DCB. The minimum DCB is 10 bytes long containing the fields DCBLNK through DCBEXT. Other fields may be added (DCBAPP) as required by the device driver. Complete definitions of the DCB fields are contained in the PSYMON Advanced Programmer's Guide.

A caller wishing to perform I/O on a specific device must perform the following steps:

1. Load the A register with any driver parameter needed. (for example, the character to be outputted)
2. Load the B register with the I/O function code. (the I/O function code is described later)
3. Load the X register with the desired DCB address.
4. Call REQIO (JSR REQIO).

The driver routine may use B, X, and Y freely without saving them, as they are saved and restored by REQIO. Register A is used for passing results and parameters. Its contents, therefore, has meaning only to the driver and the caller.

Interpretation of the various I/O function codes is also up to the device driver. The codes currently defined are as follows:

Hex code	Meaning to driver
01	Read a physical record from device
02	Write a physical record to device
04	Return device status in A register
08	Perform control function to device

Functions 01 and 02 are straightforward, being simply the traditional read and write functions. The only real difference is what constitutes a physical record. In ACIA communication with a console a physical record is a single character. I/O with a disk may define a sector as the physical record.

Function 04 returns an 8-bit status in A with the following meanings:

Bit	Meaning if bit set to 1
0	Device has input ready.
1	Device can accept output.
2	Undefined.
3	Undefined.
4	Undefined.
5	Undefined.
6	Undefined.
7	Device is inoperative or in standby.

The use of this function is dependent on the device. In an ACIA driver it might be used to test for a 'break' request, while in a disk driver it could be used to detect a write-protect condition.

The final function defined, 08, is used to perform certain non-data related control functions on a device. In the ACIA driver within PSYMON this function is used to perform the configuration functions necessary for an ACIA. Here again the function's meaning is dependent on the driver's interpretation of it.

PSYMON SUBROUTINES

One of the design goals of PSYMON was to provide a good monitor with a rich supply of useful subroutines which could be

easily used by programmers writing "system" programs. A concerted effort was made to construct useful tools that could be built upon rather than requiring the re-invention of similar functions. The subroutines discussed in this section have all been designed to be called externally. Any subroutine not mentioned here was designed for a specific purpose within PSYMON and should not be considered as a general-purpose routine. The subroutines are discussed in the order of their occurrence within PSYMON.

SEARCH - General table search.

This routine is designed to search a table of words and addresses. The word length must be fixed and is given in the first byte of the table. Addresses are two bytes long. The last byte of the table should be FF (hex). On entry register Y must point to the first byte of the item to be located in the table. Register X must point at the first byte of the table to be searched. Upon exit from this routine the Z flag, if set, indicates a successful outcome and X points to the address corresponding to the word which matched. If the Z flag is clear the item could not be located and register X points to the end sentinel of the table. Registers A and B are altered by this routine.

COMPAR - General string compare.

This routine compares two strings of arbitrary but equal length. The condition code flags are set as a result of the compare. On entry X contains the address of string 1, Y contains the address of string 2, and B contains the string length. On exit B, X, and Y are unchanged while A is altered.

LOAD - Load a hex program.

This program is designed to load a program in S1-S9 format. Input characters are obtained using the DCB pointed to by CIDCB. If CEDCB is non-zero the incoming characters will be echoed to the device whose DCB it points to. All registers are modified except U and S. The outcome of the load is reflected in the CKSUM variable in PSYMON RAM. If CKSUM is zero it indicates a successful load with an S9 termination. A non-zero value means an illegal character was encountered, a RAM error occured, or a checksum was invalid.

GETHEX - Get hexadecimal number from console.

This routine gets characters from the console (using CIDCB) to build a hexadecimal number in X. On exit A contains the last character entered (terminator), B contains a count of hex characters processed, and X contains the hex number right justified with zero fill. The Z flag is set if no hex digits were encountered, clear otherwise. Other registers are preserved.

INHEX - Input hex digit from console.

This routine inputs a character from the console (using CIDCB) and checks it for a legal hexadecimal digit. If legal the digit is converted into binary. If not the character is unchanged. The Z flag is set if the character is non-hex, clear otherwise. Registers X, Y, U, and S are unchanged.

INCHR - Input character from console.

A character is read from the console (using CIDCB) and returned in the A register. Except for C no other registers are changed. The character is stripped of parity and echoed if necessary (using CEDCB, if non-zero).

OUTCHR - Output character to console.

The character in A is output to the console (using CODCB). Only the C register is changed.

REQIO - Perform I/O request.

On entry X must point to the DCB for the device to be accessed. Register B contains the function code to be performed, while A contains a driver parameter, if required. On exit the A register may contain a driver result, depending on the function. All other registers are preserved except C.

DSPDBY - Display double byte and space.

The content of registers A and B is displayed on the console (using CODCB) as hex digits (A most significant byte) followed by a space. All registers are preserved except C.

DSPSBY - Display single byte and space.

The content of the A register is displayed on the console (using CODCB) as two hex digits followed by a space. Only the C register is altered.

OUTSP - Output a space to the console.

A single space is output to the console (using CODCB). No registers are altered except C.

OUTHEX - Output A register as 2 hex digits.

The contents of the A register are displayed on the console (using CODCB) as two hex digits. Only the C register is altered.

PSTRNG - Display string on console.

On entry X points to the string to be displayed. Characters are displayed successively (using CODCB) until a character is

encountered which has bit 7 turned on. This character is also displayed (with bit 7 masked off) and the routine exits with X pointing to the next character past the end of the string. Registers A, X, and C are changed.

CRLF - Do carriage return/line feed on console.

A carriage return and line feed are output to the console (using CODCB). Only C is altered. Note that no nulls are output following this sequence. If a device requires nulls following this sequence the device driver must provide them.

SAVE - Save a program in S1 format.

The beginning and ending addresses to be saved must be in BEGADD and ENDADD prior to calling SAVE. Output is done using CODCB. No S9 is output. This should be done by the caller if it is required. All registers are changed except U and S.

FURTHER INFORMATION

Further information regarding PSYMON may be obtained by examination of the PSYMON assembly listing. Users requiring unique modifications to PSYMON may submit their requirements to Percom Data Company for a quotation.

The above is to be presented in two parts. The first is the description of a 6809 monitor (PSYMON), that is available from PERCOM in EPROM (2716) or on disk in source, for modification or adapting to your particular needs.

Next month the entire commented source listing will be published. A study of this months article and next months article should assist those using or considering a conversion to the 6809. I can think of no better way of learning to program a new chip than to follow the example of other well coded and commented programs. A careful study of this series should assist a lot of undecided 6800 users in arrriving at a decision.

We here at 68 Micro Journal™ have made the change (we still use 6800 machines) to the 6809 and it has been a lot less painless than we first thought. Good 6809 software is coming at a much faster pace than was our experience, during the early 6800 days. It is not all here yet, but we see it coming; giving our machines more POWER and versatility than we ever imagined three or so years ago. No other chip series can equal the upgrade capacity of the 68XX series. A 6800/09 user takes a 'backseat' to no existing microcomputer today, and for a long long time to come. Also the added convenience of the 6809 has enhanced our data processing capability to a large extent. By the use of the new SWTPC 8" double sided, double density disk controller board (a complete review coming soon) and the 6809 added muscle, we (68 Micro Journal™) are able to handle our rapidly growing data files in a much more efficient manner. Used properly, with the right hardware, the 6809 puts a lot of 'minicomputers' back in the dust.

Therefore; this is published as both a mini-tutorial and as an example of 6809 code application structure and useage.

We thank Harold Mauch of PERCOM for allowing their monitor to be released, for any 6809 user to use as he or she sees fit.

Don Williams Sr.
Publisher

GIANT SOFTWARE CONTEST

Due to the the recent arrival of new and improved operating systems for the 6800/09 machines, we have decided to sponsor a software contest, effective with this issue.

We will award prizes in a number of categories.

Prizes for each category will be:

FIRST - Life Subscription 68 Micro Journal"

SECOND - 6 year extension 68 Micro Journal"

THIRD - 3 year extension 68 Micro Journal"

4th-10th 1 year extension 68 Micro Journal"

The software must be utilities or serious software, of original design, to operate with the following CATEGORIES:

TSC FLEX	6800 Disk System MiniFlex
TSC FLEX	6800 Disk System FLEX Ver. 2.0 5"
TSC FLEX	6800 Disk System Ver. 1.0 8"
TSC FLEX	6809 Disk System Ver. 09 5" or 8"
SSB	Dos Version 5
SSB	Dos Version 4 or earlier
SSB	TSC FLEX Version
PERCOM	INDEX
PERCOM	MiniDos+
NEMENWAY	CP/68 Disk System
MSI	Disk Operating Systems
SOFTWARE DYNAMICS	SDOS
JPC	TC-3 Cassette System
ANY	KC Standard Tape System
BASIC	Any 6800 Version
BASIC	Any 6809 Version

There are sixteen (16) categories, as indicated above. This means there will be ONE HUNDRED AND SIXTY (160) total prizes guaranteed. In addition we hope to have other prizes donated by various vendors of 6800/09 products. As these materialize we will publish a list of any additional prizes.

Final decision shall be delegated to a panel of judges from the staff of 68 Micro Journal". All judges decisions are final and each person submitting, shall by his or her submitting material for evaluation, acknowledge that they agree to abide by any and all rules of this contest, as published within the pages of 68 Micro Journal".

Programs and material submitted shall be judged on the basis of good and workable software. By this we mean, it should do something useful and be needed by the average 6800/09 user in the particular category. Size is of little importance, the most important consideration will be how useful it is.

All material submitted shall remain the property of the original owner (who should be the author). Each submission shall contain a paragraph that states the material submitted is of original design and the property of the person in who's name it is submitted.

It shall be understood that regardless of who wins or does not win a prize, all material submitted shall be authorized and eligible, to be published by 68 Micro Journal". Material published, which was not a winning entry, shall gain the author an extension to his or her subscription. Anyone may enter and it is not a requirement that the person submitting material be a subscriber to 68 Micro Journal". Prizes will be awarded on the quality of the material submitted and being or not being a subscriber, will have no bearing.

Authors should indicate that the material has NOT been previously published in any commercial magazine or journal (club newsletters and the like do not count as a commercial magazine or journal).

I have tried to keep the rules simple. This should encourage the maximum participation in the contest. This is another of the ways that we attempt to secure good material for the sole benefit of our readers. Also I believe that it will encourage those who have developed good software, to share with his or her fellow 6800/09 users. By sharing we all profit. By working together, as has been in the past, it enables us as 6800/09 user to have a magazine that is just for us.

If you do not believe this, just look in ALL the others, among them articles for everything; everthing that is except the 6800/09.

This contest will close February 15th, 1980. This will be the first anniversary of 68 Micro Journal". Prize winners will be announced in the April 1980 issue of 68 Micro Journal".

6809

```
                        *
                        * PATCHES TO NEWDISK COMMAND
                        * FOR FORMATTING 40 TRACKS
                        *
                        * CAUTION DO NOT USE WITH
                        * SHUGART 35 TRACK DRIVES.
                        * USE ONLY WITH 40 TRACK
                        * DRIVES SUCH AS WANGCO.
                        *
                        * WRITTEN BY JOHN BYRNS
                        *  21-AUG-79
                        *
             0028       TRACKS  EQU     40
                        *
C1ED                            ORG     $C1ED
C1ED 28                         FCB     TRACKS
                        *
C263                            ORG     $C263
C263 28                         FCB     TRACKS
                        *
C2C0                            ORG     $C2C0
C2C0 27                         FCB     TRACKS-1
                        *
C32F                            ORG     $C32F
C32F 28                         FCB     TRACKS
                        *
C3EC                            ORG     $C3EC
C3EC 27                         FCB     TRACKS-1
                        *
                                END

0 ERROR(S) DETECTED
```

MICROWARE ABASIC COMPILER

A 68 Micro Journal Lab Review

The run-time of a compiled Basic program is usually much shorter than running the same program with a Basic interpreter. But compilers are often more difficult to use than interpreters, and so often the total time to write, debug, and get a program running is longer with a compiler than with an interpreter.

This is a problem which Microware Systems Corp. has attacked (and partially solved) with their A/BASIC. The complete A/BASIC system consists of a compiler, an interpreter, and a source generator.

The compiler is available in several versions. The original cassette A/BASIC compiler costs $65, and requires several cassette read passes to compile a program. Microware now also has disk versions for the SWTP and SSB disk systems, which greatly simplify the job of compiling. These versions also handle sequential disk files, and cost $150. A Percom disk version is also available. In addition, Percom has a patch which adapts the cassette version to run on the Percom disk; this version does not support disk files.

To simplify the job of debugging a program, Microware has recently introduced an A/BASIC interpreter which is compatible with the compiler. This makes it possible to write a program, run and interactively debug it with the interpreter, and once the program is bug-free, compile it into a very fast machine language version.

The A/BASIC Source Generator is an add-on to the disk versions of the compiler, which prints out the assembly-language listing of the compiled object program, and also prepares an assembly language source file on disk. This file can then be manually edited or optimized, and reassembled into a possibly even faster or more powerful machine language program.

For this review, we looked at cassette version 1.0C (as modified by the Percom patch), and at SWTP disk version 2.0F, which runs under the mini-Flex operating system. Let's answer some common questions.

Is A/BASIC Fast?

Here are some results on three different programs:

KILOBAUD Benchmark Program 7
(from October 1977 KILOBAUD)

SWTP Disk Basic v. 3.0	190 secs
Altair 680 (Microsoft) Basic	42 secs
TSC Basic (FLEX 2.0 version)	30 secs
SD Basic Compiler	24 secs
A/BASIC (both versions)	3 secs

All of these were run on the same system with a 1 MHz clock speed, except for the SD Basic; its timing came from the review in the July 1979 issue of '68' Micro., and was recomputed for the same clock speed.

This is not really a fair comparison, since A/BASIC is integer only, whereas the others are floating-point. So we ran another test, this time playing ELIZA. Starting with the same Basic version of ELIZA (except that some changes had to be made to adapt the Basic source code to each particular Basic system):

ELIZA Benchmark

SWTP Disk Basic v. 3.0	80 secs
TSC Basic (FLEX 2.0 version)	6 secs
A/BASIC (both versions)	2 secs

In each case, we timed how long it took for ELIZA to come up with an answer to a particular input statement. This test was an interesting one, because it showed how fast TSC Basic is with purely string operations. Keep in mind that on strings, A/BASIC is at its worst, since it uses string subroutines like an interpreter, rather than in-line machine code.

A third test was run, this time on a program which reads a disk input file, searches for strings, saves them in a table, and does a memory sort. Again, the Basic program was the same, and the data file was the same.

Disk Benchmark

SWTP Disk Basic v. 3.0	253 secs
TSC Basic (Flex 2.0 version)	29 secs
A/BASIC (SWTP disk version)	7 secs

This program was not as heavily string oriented, and so A/BASIC came out a bit better. (Nevertheless, the times turned in by TSC Basic are astounding too.)

How Do You Use It?

The A/BASIC compiler is much easier to use than other microprocessor compilers. Most other compilers produce an assembly language output; this must then be assembled into a relocatable object file, which is then loaded into memory by a linking loader, which also loads in any subroutines needed. These subroutines are usually called a 'run-time package'. Many compiled programs wind up with machine language code which is nothing but a series of calls to subroutines, which do the actual work. The job of putting all this together requires quite a bit of work on the

users part, as he calls in one program after another.

A/BASIC is quite different, in that it produces a machine language program directly. Although Microware claims that no run-time package is used, this is not strictly true. There are some run-time subroutines that are added to the machine code, but these are added in directly by the compiler during the compilation. No separate linking loader is required, so these subroutines are not really visible to the user.

With a disk system, compiling an A/BASIC program is very easy (we didn't try using just cassettes.) You simply prepare a text file with the editor and put it on disk, call A/BASIC, and give it the information as to where to get the Basic text and where to put the machine language program. As soon as the compiler is done, you can load the resulting program into memory and run it. No other steps are needed.

The cassette version with the Percom disk patches worked flawlessly; the mini-Flex version had an occasional tendency to die if the source program had an error which the compiler didn't know how to treat. Since this version is quite new, we expect future revisions to reduce this problem.

In addition to the Basic code, the source program needs additional compiler directives before compiling. These are an OPTION statement which is similar to that of some assemblers; and an ORG and BASE statement. The ORG is also like the ORG of an assembler, and specifies the starting memory address for the program. The BASE is similar, but specifies the starting address for variables and arrays. Either of these can be redefined during a program, so that the program can be compiled for any specific location in memory, or even broken up into sections. This also makes it easy to share common variables between programs, simply by making sure to assign compatible addresses.

The compiled program can be located anywhere in memory, but in all cases, A/BASIC uses locations 0020-002F as a scratchpad.

When a Basic program is interpreted, the interpreter checks the program not only for syntax errors, but also for certain kinds of logical errors. For instance, it will not let you use an array larger than has been dimensioned. But a compiler does not do any error checking once the program has been compiled and is running (unless you insert ON ERROR statements into the source.) Thus it is very easy for the compiled program to bomb if it has mistakes.

This is a problem with any microcomputer compiler (some large systems do check for run-time errors, but most microcomputers don't) and makes debugging compiled programs somewhat of a nuisance. Since any reasonably sized program has at least several mistakes, it may take a number of passes before it is running properly. With an interpreter, it's easy to go back and forth, modifying a program and running it immediately to see what happens. With a compiler, this is a much longer process. Loading the editor, changing the text, putting it back on disk, loading and executing the compiler, and then loading the object program and executing can take five or ten minutes each time you do it. (Here's where the fast speed of the Percom disk made the job much more pleasant that the long wait for the mini-FLEX DOS to load and save files.)

A/BASIC is faster in this respect than other compilers, since the compiled program is ready to run and does not have to be linked to a run-time package. Nevertheless, it still takes a lot longer than when using an interpreter. Microware's A/BASIC interpreter is intended to solve this problem, by allowing you to debug a program and get it fully working with the convenience of an interpreter, yet compile it for fast speed later.

Debugging a compiled program is also done in a completely different way. SWTP Basics allow tracing of a program, stopping at any point to examine or change variables, and then continuing. This can't be done as readily in a compiled program.

A compiled machine language program must be debugged the same way an assembly language program would be debugged. As part of the compilation, A/BASIC produces a printout which shows the location in memory of every variable, as well as the address where each Basic statement has been translated into. An optional printout gives the full machine language output in the printout, so each part of a statement can be located in memory. (The machine language printout is produced directly by the compiler; Microware's A/BASIC Source Generator produces an assembly language printout.) Using this listing, you can use the monitor's Breakpoint function to step through the program to see what is happening, or even use a tracer program (available from Ed Smith, SSB, or TSC.)

A/BASIC Language Features

The A/BASIC language is somewhat different from the Basic used by the SWTP, TSC, or Computerware Basic interpreters.

First of all, A/BASIC is strictly an integer Basic. It can handle integers in the range of -32768 to +32767; it uses two bytes per number with two's complement notation. It will also accept unsigned numbers up to 65535 for use in some functions (although numbers greater than 32767 may be treated as negative in some cases.) It also accepts hexadecimal numbers up to $FFFF.

(But since the Carry bit in the CC register can be tested with an ON OVERFLOW GOTO statement, it's easy to program multiple precision arithmetic.)

Because of this, it does not have some of the common functions such as SIN or SQR. The basic arithmetic functions are ABS and RND. But RND returns an integer rather than a fraction, so programs using RND may have to be changed slightly. There is also no ^ operator for exponentiation.

In the cassette version, all strings are 32 characters long; the disk version defaults to 32 character strings, but a string may be dimensioned anywhere from 1 to 255 characters long. Within one program, each string can be a different length, a tremendous memory saver in many applications. All the common string functions like MID$ or LEFT$ are present, as well several new ones. TRM$ is used to trim blanks from the end of a string; SUBSTR is a substring search function which searches one string for occurrences of another. This one function can replace an entire loop in other Basics.

A/BASIC lets the user access the I/O buffer used in INPUT or PRINT statements directly; it is called BUF$, and can be used the same as any other string variable. This is useful, for instance, in reading strings which have embedded commas; the commas would normally break up the string.

Two-dimensional numeric arrays and one-dimensional string arrays are possible. The DIM statement can also change the size of a string variable or array element.

The disk version of A/BASIC has all the standard disk file I/O statements, including OPEN, CLOSE, READ and WRITE, CHAIN, RESTORE, and SCRATCH. One welcome change from SWTP Disk Basic is that the OPEN statement can have a variable for the file-name, rather than requiring an actual file name spelled out.

For business or conventional programs (whatever that is), A/BASIC has some limitations aside from its lack of floating point. It has no DATA statement. The cassette version requires a numerical address in PEEK and POKE statements, and will not accept a variable as in PEEK(X). Strings can be compared for equality or non-equality, but not sorted; if you needed something like IF A$<B$, you would have to break up each string into characters, convert them into numbers with the ASC function, and do a numeric compare, character by character. Fortunately, A/BASIC is fast enough that this would make little difference in the running time.

Another limitation is how A/BASIC handles the IF statement. Other 6800 Basics allows the IF to be followed by any valid statement, such as in IF P=1 THEN PRINT X. In A/BASIC, only a transfer is allowed, and then only in the form of

IF ... THEN <line number>, or
IF ... GOSUB <line number>.

But A/BASIC shines when it comes to writing the kinds of programs which would normally be done in assembly language. The disk version is especially convenient here, since it makes it easy to use disk I/O, a job which would normally be messy in assembly language.

There are a lot of A/BASIC features which are specially aimed at the assembly language programmer.

A/BASIC supports logical operations on numeric variables, including AND, OR, exclusive OR, and inverting (NOT).

A CALL statement allows a machine language subroutine to be directly called from A/BASIC; since the accumulators generally have the last number worked on at the end of a statement, this allows the transfer to a subroutine with an argument in the accumulators. For instance, the sequence

```
Z= 3*I - 7
CALL $B000
```

would go to a subroutine at location $B000 with the 16-bit value of Z in the two accumulators.

A GEN statement allows data or machine language instructions to be directly inserted into a program.

IRQ ON and IRQ OFF statements control the interrupt mask bit in the CC register; RETI inserts an RTI machine instruction into the code.

Both versions of A/BASIC will run with Microware's RT-68MX monitor; this is a MIKBUG (tm) replacement which allows multi-tasking. Thus A/BASIC has commands to control task switching of RT-68MX.

The cassette version requires the RT-68 monitor, and would need extensive patching to adapt to other monitors; the disk based version will run with any monitor supported by the DOS. (In any case, all the multi-tasking functions of A/BASIC require

the RT-68 monitor, and cannot be used with other monitors.)

'Running the cassette-based compiler requires two cassette machines with motor control.

Conclusions

Microware's A/BASIC, especially in the disk version, is a quite powerful and very easy to use compiler. It provides an easy way to generate fast, compact programs which would have been difficult to write in some other language. Due to its lack of floating point, it may not be as popular among the scientific or business users, but for industrial control, non-numeric processing, or just plain games, it's excellent.

Additional information may be obtained from:
Microware Systems Corp.
P.O. Box 4865
Des Moines, IA 50304

A 68 Micro Journal rating: AAA

Rating scale:
AAA - Excellent
AA - Good
A - Fair (could be better but works)
P - Poor and may not always work properly
X - Not recommented for children (or anything else).

LINE EDITOR

Dr. Chuck Adams
421 Frankie Ln.
Lewisville, TX 75067

The program listing 1 is a 6800 assembly language line editor. The editor was created as a "front-end" processor for a 3K resident two-pass assembler. An earlier version of the editor was written using a linked-list for memory storage of text, but after some thought was abandoned for the listing enclosed. In storing each line with a two byte pointer for the linked-list, considerable storage was consumed for many lines of source and thus the procedure was replaced with the new structure.

Each line of text is entered into memory with a two byte line number, initially created by the editor when new text is entered, followed by the line of text and terminated with a carriage return. The line number is a two byte unsigned binary integer, thus allowing 65,535 non-zero line numbers, i.e. if there was sufficient memory, which there is obiously not, since it would require 128K of memory just for the storage of all two byte combinations.

Since most text contains a significant number of blanks, especially in assembly language program source, blanks are stored as a one byte count within a string of text. Each non-blank character is stored with high-order bit on, thus the largest blank count embedded within a line is 127, not too serious a constraint upon the programmer.

The editor commands are simple one character commands, or one character command followed by one or more optional parameters. These commands are easy to remember and simple to understand.

COMMANDS

N NEW - Create new string of text in memory.

CNTL-C CANCEL - Cancel entry mode of text.

P(cr) PRINT - Print text starting at beginning.

P N(cr) PRINT - Print text starting at line #N.

D N(cr) DELETE - Delete line numbered N.

D N,M(cr) DELETE - Delete lines numbered N thru M.

I N(cr) INSERT - Insert new line into text with number N

S SAVE - Save text to tape. Starts when key pressed on keyboard.

L LOAD - Load text from tape.

E EXIT - Exit editor and return to monitor.

? QUERY - Print value of last memory location used. May be necessary for small system

DOUBLE DENSITY

The most reliable, cost effective disk system ever designed for the SS-50 bus is now available. The Southwest Technical Products Corp. DMF-2 disk system provides 2.5 M/bytes of usable (formatted) on-line storage. It offers the lowest cost per byte available on floppy disks at this time.

The DMF-2 features "Qume" DATATRAK 8 double headed eight-inch drives. We consider these to be the best drives we have ever tested. The 17½" x 5" x 21½" cabinet is made from 1/8 inch thick aluminum and finished with a super tough textured epoxy. The power supply has 115/230 volt capability and will operate from either 50 or 60 Hz. mains.

The controller is a direct memory access type circuit, using the 6844 DMA controller and a 1791 double density disk controller. This type circuit has a much higher data transfer rate than simple sector buffer type circuits and it also imposes far less overhead on the processor. The critical phase lock and data separator circuits use 1% components and time proven circuits to insure long term reliability. We find no statistical difference in the error rate of this controller and our single density controller.

The DMF-2 is supplied with the FLEX®-09 operating system. You can format and record in either single or double density. FLEX® is the world standard disk operating system for the MC6809 and is available for almost all 6809 family hardware, whatever the source.

The DMF-2 system includes the cabinet, power supply, controller, connecting cable, diskette with FLEX®-09, two drives and instruction manual. Shipping weight is 53 pounds.

DMF-2 Dual Double Density Disk Driver—assembled and tested $2,495.00
DMF-2 Controller board retrofit for converting DMF-1 single density systems . . . . . $ 395.00

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Print with Quality and Speed

The Southwest Technical Products fast quality printer system is based on the "Qume" Sprint 3/45 daisy-wheel printer. For word processing applications, where quality and speed are both necessary , this printer is the answer. Over a hundred styles of printer wheels are available, including proportional space and foreign type fonts.

The SP-3 is supplied with the following features: out of paper detect, out of ribbon detect, top of forms eject, bottom feed slot, cover interlock, operator lights, paper handling system and switching power supply. Optional forms tractors are available for applications where these are desired. The SP-3 printer is supplied with a twelve-line interface and connecting cable for use with all Southwest Technical Products computers.

- Average text print speed of 45 characters/second
- Prints full characters of electric typewriter quality
- Uses variable intensity ballistic hammer which automatically adjusts to correct one of six strike intensities according to character size
- Accepts single sheets and continuous forms, with or without sprocket holes
- Prints on forms up to 15 inches wide
- 96 character positions on "daisy" printwheel
- Wide variety of standard font styles available in 10 and 12 pitch and proportional spacing
- Prints 132 columns at 10 characters/inch
- Prints 158 columns at 12 characters/inch
- Prints proportional spacing in increments of 1/120 inch, left or right
- Features electronic tabbing and carriage return up to 13.1 inches at 320 ms maximum
- Vertical spacing in increments of 1/48 inch, up or down
- Vertical slew rate of 5 inches per second
- Plotting resolution of 5760 points per square inch
- Features pressure platen; pin feed platens optional
- Easy to handle ribbon cartridge with multi-strike carbon, single strike carbon, or fabric ribbon available in black and colors
- Printwheel is easily operator changeable
- Operator controls include horizontal forms positioning, vertical forms positioning, forms thickness and ribbon advance

SP-3	Daisywheel Printer—with listed features, interface and power supply	$2,995.00
SP-5	Serial Daisywheel Printer—with above features and power supply (less serial interface)	$3,195.00
80026-01	Optional Forms Tractor	$ 190.00

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

to check for remaining memory.

The program has been throughly tested and been in use by the author for several months. No errors have been found, but the author would appreciate any comments on any that may still remain. Many other commands may be added and the author again would appreciate any ideas or implementations added by the reader. No attempt has been made to optimize the code any further and the reader may spend many hours of productive time doing so, but the author feels that the readers time may be better spent on other projects.

The output routines require the use of an ACIA at locations $8004 and $8005 in order to allow the user to interact with the editor when a listing operation is being performed. If the user does not have such an interface on his or her computer, then these sections may be replaced with NOP's.

The program listing is from the resident version of the editor and a 3K assembler on a Southwest Technical Products Co. MP68 system with 32K of memory. The source program text for the editor requires only 12K of memory for storage, thus large programs may be created at a speed of assembly of over 100 lines of code per second (without listing the output) and object code generated for tape or memory storage. This system may be of benifit to both the hobbyist and professional programmer for development of 6800 software.

```
                        *       SIMPLE LINE EDITOR
                        *       COPYRIGHT (C) 1978 - ALL RIGHTS RESERVED
                        *
                        *
                        *       DR. CHUCK ADAMS
                        *       DEPARTMENT OF COMPUTER SCIENCES
                        *       NORTH TEXAS STATE UNIVERSITY
                        *       DENTON, TEXAS  76203
                        *       (8171 788-2762  MONDAY-FRIDAY
                        *
                        *
                        *
                        *       PAGE 2600 DEFINITIONS
                        *
0020                    ORG     $20
0020 7E E1AC GETC       JMP     INEEE   GET CHAR FROM TERMINAL
0023 7E E1D1 OUTC       JMP     OUTEEE  OUTPUT CHAR TO TERMINAL
0026                    BEGIN   RMB     2       STARTING ADR OF BLOCK TO BE MOVED
0028                    END     RMB     2
002A                    MEM1    RMB     2       DESTINATION ADR OF BLOCK
002C                    MEM2    RMB     2       CALCULATED ENDING ADDRESS
002E                    SAVE    RMB     2       SAVE AREA FOR STACK POINTER
0030                    LINEMB  RMB     2       RESERVE TWO BYTES FOR CURRENT LINE NUMBER
0032    1000 STRMEM     FDB     $1000   ADDRESS FOR START OF MEMORY BUFFER AREA
0034                    ERRMEM  RMB     2
0036                    BACKCK  RMB     2       TWO BYTES FOR BUFFER OVERRUN CHECK
0038                    TEMPNR  RMB     2       TEMPORARY AREA
003A                    NEXTBG  RMB     2       NEXT AVAILABLE LINE NUMBER
003C                    DEL1    RMB     2       FIRST NUMBER TO DELETE
003E                    DEL2    RMB     2       SECOND NUMBER TO DELETE
0040    00      NROUT   FCB     0       FLAG FOR A PROMPT
0041 86 0D      CRLF    LDAA    #$0D    LOAD CR
0043 BD DE              JSR     OUTC    OUTPUT CARRIAGE RETURN
0045 86 0A              LDAA    #$0A    LOAD LF
0047 BD DA              JSR     OUTC    OUTPUT LINE FEED
0049 B6 8004            LDAA    ACIAS   GET ACIA STATUS
004C 47                 ASRA            SEE IF INCOMING CHARACTER
004D 24 10              BCC     DEII    NO CDIIEEB
004F B6 8005            LDAA    ACIAD   GET 'EM COMDDF
0052 84 7F              ANDA    #$7F    REMOVE PARITY
0054 81 03              CMPA    #$03    SEE IF CANCEL COMMAND
0056 26 03              BNE     CAN2    NO CANCEL, MUST BE USER
0058 7E 0100 CAN1       JMP     START   WAS CANCEL
005B B6 8004 CAN2       LDAA    ACIAS   WAIT FOR CRITTER
005E 47                 ASRA            LOOK AT STATUS
005F 24 FA              BCC     CAN2    LOOP TYPIST, WAIT
0061 B6 8005            LDAA    ACIAD   GET CHAR
0064 84 7F              ANDA    #$7F    PLEASE, NO PARITY
0066 81 03              CMPA    #$3     SEE IF CANCEL
0068 27 EE              BEQ     CAN1    YEP, THAT'S ALL
006A 39         RETI    RTS             RETURN
                        *
006B 8D B4      PROMPT  BSR     CRLF    OUTPUT CR/LF
006D B6 3E              LDAA    #'>     GET PROMPT CHARACTER
006F 20 02              BRA     OUTC    OUTPUT PROMPT
0071 B6 20      OUTSP   LDAA    #$20    LOAD BLANK
0073 20 A8              BRA     OUTC    OUTPUT AND RETURN TO CALLING ROUTINE
0075            BLKCNT  RMB     1
0076            SAVEX   RMB     2       SAVE AREA FOR IX
0078            SAVEA   RMB     1       SAVE AREA FOR ACCA
0079    03E8 K10K       FDB     1000
007B    0064 HUND       FDB     100     HUNDRED
007D    000A TEN        FDB     10      TEN
007F    0001 ONE        FDB     1
0081            BUFFER  RMB     80      BUFFER FOR INSERT
00D1            BUFEND  RMB     2       POINTER TO LAST OF BUFFER
                        *
                        *
                        *
0100                    ORG     $100
0100 CE 0128 START      LDX     #BTABLE GET START OF BRANCH TABLE
0103 BD 006B            JSR     PROMPT  OUTPUT PROMPT
0106 BD 0020            JSR     GETC    GET INPUT COMMAND CHARACTER
0109 A1 00      LOOP    CMPA    0,X     COMPARE INPUT CHARACTER WITH CHAR IN TABLE
010B 26 04              BNE     SKIP    SKIP TO NEXT COMMAND IN TABLE
010D EE 01              LDX     1,X     MATCH WAS FOUND MOVE TO ADDRESS
010F AE 00              JMP     0,X     GOTO ROUTINE
                        *
                        *
0111 08         SKIP    INX             GO ONE BYTE PAST CHAR IN TAB
0112 08                 INX             BYPASS FIRST BYTE OF ADDRESS
0113 08                 INX             BYPASS SECOND BYTE OF ADDRESS
0114 8C 013E            CPX     #TABEND MAKE SURE WE DON'T GET INTO TWILIGHT ZONE
0117 26 F0              BNE     LOOP    CHECK ON NEXT CHARACTER IN TABLE
                        *
                        *       LET USER KNOW CHAR NOT FOUND
                        *
0119 86 3F      ERIP1   LDAA    #'?     LOAD QUESTION MARK
011B BD 0023            JSR     OUTC    OUTPUT QUESTION MARK
011E 20 E0              BRA     START   GO BACK TO COMMAND PROCESSOR
                        *
                        *
                        *
0120    4E      BTABLE  FCC     'N'     FOR NEW FILE
0121    013E            FDB     NEW     ADDRESS FOR NEW ROUTINE
0123    49              FCC     'I'     FOR INSERT
0124    0612            FDB     INSERT
0126    44              FCC     'D'     FOR DELETE
0127    02AB            FDB     DELETE
0129    50              FCC     'P'     FOR PRINT
012A    024F            FDB     PRINTS
012C    45              FCC     'E'     FOR EXIT TO MONITOR
012D    C000            FDB     $E000
012F    3F              FCC     '?'     QUERY FOR LAST ADDRESS
0130    0102            FDB     QUERY
0132    52              FCC     'R'     FOR RENUMBER
0133    037E            FDB     RENUM
0135    43              FCC     'C'     FOR CONTINUE
0136    03A1            FDB     CONT
0138    53              FCC     'S'     SAVE TO TAPE
0139    03F0            FDB     SAVTPE
013B    4C              FCC     'L'     LOAD TAPE
013C    0400            FDB     LODTPE
                        TABEND  EQU     *
                        *
                        *
                        *
013E 8E 27      NEW     LDS     STRMEM  STARTING ADDRESS
0140 7F 0075            CLR     BLKCNT  ZERO OUT BLANK COUNTER
0143 7F 0030            CLR     LINEMB  CLEAR LINE NUMBER
0146 B6 0A              LDAA    #$0A    BINARY 10
0148 97 31              STAA    LINEMB+1 SET LINE NUMBER TO 10
014A 4F 00              CLR     0,X     STORE INTO MEMORY
014C A7 01              STAA    1,X
014E 7D 0046 NEW10      TST     NROUT   SEE IF INPUT FROM TAPE
0151 26 0C              BNE     NEW11
0153 BD 0041            JSR     CRLF    OUTPUT CR/LF
0156 BD 024A            JSR     GETLNR  GET LINE NUMBER FOR OUTPUT
0159 BD 0220            JSR     CVBTD   OUTPUT SAVE
015C BD 0071            JSR     OUTSP   OUTPUT SPACE
015F 08         NEW11   INX             ADVE POINTER PAST LINE NUMBER
0160 DF 36              STX     BACKCK  STORE IX FOR BACKSPACE OVERRUN
0162 08                 INX
0163 BD 0020 INCH       JSR     GETC    GET CHAR FROM TERMINAL
0166 81 0A              CMPA    #$0A
0168 27 F9              BEQ     INCH
016A 81 03              CMPA    #$03    CHK FOR CANCEL FILE CREATE
016C 27 40              BEQ     ENDUP   IF SO, CLEAN UP
016E 81 08              CMPA    #$08    CHECK FOR BACKSPACE
0170 26 15              BNE     INCH0   NO, CONTINUE
0172 7A 0075            DEC     BLKCNT  DEC BLANK COUNT
0175 2C EC              BGE     INCH    MORE BLANKS
0177 7F 0075            CLR     BLKCNT  ZERO OUT, NO NEGATIVE ALLOWED
017A 09                 DEX
017B 9C 36              CPX     BACKCK  SEE IF ABOUT TO OVERRUN NUMBER
017D 27 02              BEQ     INCH0   YES, HERE
017F 20 E2              BRA     INCH    NO, DO OK
0181 08         INCH0   INX
0182 BD 0071            JSR     OUTSP   OUTPUT SPACE, MOVE CURSOR BACK TO START OF LINE
0185 20 DC              BRA     INCH    HERE?
0187 81 20      INCH1   CMPA    #$20    SEE IF BLANK
0189 26 05              BNE     INCH2   NO, OTHER CRITTER
018B 7C 0075            INC     BLKCNT  INC BLANK COUNT
018E 20 D3              BRA     INCH    NEXT?
0190 8A 80      INCH2   ORAA    #$80    TURN ON HIGH ORDER BIT
0192 B6 75              LDAB    BLKCNT  GET BLANK COUNT
```

```
0194 27 06          BEQ   INCH3    NO, STORE CRITTER
0196 E7 00          STAB  0,X      STORE BLANK COUNT
0198 08             INX            MOVE TO NEXT LOCATION
0199 7F 0075        CLR   BLKCNT   START OVER ON COUNT
019C A7 00   INCH3  STAA  0,X      STORE CRITTER
019E 08      *      INX            MOVE ALONG
019F 81 0D          CMPA  #$0D     CHECK FOR CR
01A1 27 02          BEQ   INCH4    CREATE NEXT LINE
01A3 20 DE          BRA   INCH     NEXT?
01A5 BD 01CB INCH4  JSR   INCLIN   INC LINE NR BY 10
01A8 A7 00          STAA  0,X      STORE BASE
01AA E7 01          STAB  1,X
01AC 20 A0          BRA   NEW10
             *
01AE DE 36   ENDUP  LDX   BACKCK   GET LAST BACK CHECK
01B0 09             DEX
01B1 DF 34          STX   ENDMEM   END END OF MEMORY
             *
01B3 EE 00          LDX   0,X
01B5 DF 3A          STX   NEXTBD
01B7 DE 34          LDX   ENDMEM
01B9 6F 00          CLR   0,X      END EOF IN STREAM
01BB 6F 01          CLR   1,X
01BD 86 15          LDAA  #$15     RESTORE ACIA
01BF B7 8004        STAA  $8004    ACIA CR
01C2 7F 0075        CLR   BLKCNT
01C5 7F 0040        CLR   NROUT
01C8 7E 0100        JMP   START
             *
             *
01CB D6 31   INCLIN LDAB  LINENR+1 GET LOW BYTE OF LINE NO
01CD CB 0A          ADDB  #$0A     ADD 10
01CF D7 31          STAB  LINENR+1 SAV RESULT
01D1 24 03          BCC   INCLI1   NO CARRY
01D3 7C 0030        INC   LINENR   ADD CARRY
01D6 96 30   INCLI1 LDAA  LINENR
01D8 39             RTS
             *
01D9 9F 3E   BLOCK  STS   SAVE     SAVE STACK IN SAVE AREA
01DB 0F             SEI
01DC D6 29          LDAB  END+1    GET END ADD
01DE D0 27          SUBB  BEGIN+1  CALCULATE END LENGTH
01E0 96 28          LDAA  END
01E2 92 26          SBCA  BEGIN
01E4 24 01          BCC   BLOCK1   NO ERROR
01E6 3F             SWI            FLAG AND HALT
01E7 DB 2B   BLOCK1 ADDB  NEW1+1   CALCULATE NEW ENDING ADR
01E9 99 2A          ADCA  NEW1
01EB 97 2C          STAA  NEND     STORE END ADR
01ED D7 2D          STAB  NEND+1
01EF 96 26   TEST1  LDAA  BEGIN    GET ADR
01F1 91 2A          CMPA  NEW1     CHECK HIGH ORDER BYTE
01F3 22 19          BHI   BACK     MOVE TO LOW MEMORY
01F5 26 06          BNE   FRONT    MOVE TO HIGH MEMORY
01F7 96 27   TEST2  LDAA  BEGIN+1
01F9 91 2B          CMPA  NEW1+1
01FB 22 11          BHI   BACK
01FD DE 28   FRONT  LDX   END
01FF 9E 2C          LDS   NEND     GET END ADD
0201 08             INX            SET IX UP
0202 09      LOOP1  DEX            POINT TO BYTE
0203 A6 00          LDAA  0,X      GET BYTE
0205 36             PSHA           STORE IN NEW AREA
0206 9C 26          CPX   BEGIN    THRU?
0208 26 F8          BNE   LOOP1    NO
020A 9E 2E          LDS   SAVE     RESTORE STACK
020C 0E             CLI
020D 39             RTS
020E DE 2A   BACK   LDX   NEW1     GET START OF NEW BLOCK
0210 9E 26          LDS   BEGIN    ADR OF OLD BLOCK
0212 34             DES            SET UP STACK
0213 09             DEX            SET UP IX
0214 08      LOOP2  INX            POINT TO STORAGE
0215 32             PULA           GET CHARACTER
0216 A7 00          STAA  0,X      STORE
0218 9C 2C          CPX   NEND     THRU?
021A 26 F8          BNE   LOOP2    NO
021C 9E 2E          LDS   SAVE
021E 0E             CLI
021F 39             RTS
             *
0220 DF 76   CVBTD  STX   SAVEX    SAVE IX
0222 CE 0079        LDX   #K10K    POINT TO TABLE
0225 7F 0078 CVDEC1 CLR   SAVEA    ZERO OUT COUNT
0228 E0 01   CVDEC2 SUBB  1,X      SUBTRACT
022A A2 00          SBCA  0,X      DITTO
022C 25 05          BCS   CVDEC5   OVERFLOW, RESTORE
022E 7C 0078        INC   SAVEA    END ANOTHER FACTOR
0231 20 F5          BRA   CVDEC2
0233 EB 01   CVDEC5 ADDB  1,X
0235 A9 00          ADCA  0,X      ADD HIGH ORDER BYTE
0237 36             PSHA           SAVE RESTORED RESULT
0238 96 78          LDAA  SAVEA    GET COUNT
023A 8B 30          ADDA  #$30     CREATE ASCII
023C BD 0023        JSR   OUTC     OUTPUT CRITTER
023F 32             PULA           RETRIEVE PARTIAL RESULT
0240 08             INX
0241 08             INX
0242 8C 0081        CPX   #K10K+8  SEE IF THROUGH
0245 26 DE          BNE   CVDEC1   ONE OR MORE REMAINING
0247 DE 76          LDX   SAVEX    RESTORE IX
0249 39             RTS
             *
             *
024A 96 30   GETLNR LDAA  LINENR
024C D6 31          LDAB  LINENR+1
024E 39             RTS
             *
             *
024F DE 32   PRINTS LDX   STRMEM   GET START
0251 BD 0071        JSR   OUTSP    OUT SPACE
0254 BD 0329        JSR   INNR     GET LINE #
0257 96 38          LDAA  TEMPNR
0259 D6 39          LDAB  TEMPNR+1
025B BD 0309 PRINT6 JSR   NIBCMP
025E 2F 07          BLE   PRINT    FOUND FIRST LINE SO START PRINT
0260 36             PSHA
0261 BD 02CA        JSR   DELE2
0264 32             PULA
0265 20 F4          BRA   PRINT6
0267 BD 0041 PRINT  JSR   CRLF
026A 7D 0040        TST   NROUT
026D 26 0A          BNE   PRINT0
026F A6 00          LDAA  0,X      GET LINE NUMBER
0271 E6 01          LDAB  1,X
0273 BD 0220        JSR   CVBTD    OUTPUT LINE NUMBER
0276 BD 0071        JSR   OUTSP    OUTPUT SPACE
        0279 PRINT0 EQU   *
0279 08             INX
027A 08             INX
027B A6 00   PRINT1 LDAA  0,X      GET CHAR FROM BUFFER
027D 2B 0C          BMI   PRINT3
027F 16             TAB            MOVE BLANK COUNT TO ACCB
0280 86 20          LDAA  #$20     LOAD BLANK
0282 BD 0023 PRINT2 JSR   OUTC     OUTPUT BLANK
0285 5A             DECB           DEC COUNT
0286 26 FA          BNE   PRINT2
0288 08             INX            POINT TO NEXT CRITTER
0289 20 F0          BRA   PRINT1
028B 84 7F   PRINT3 ANDA  #$7F     STRIP OFF HIGH BIT FOR PRINTING
028D 81 0D          CMPA  #$0D     SEE IF CR
028F 27 06          BEQ   PRINT4   SPECIAL HANDLING
0291 BD 0023        JSR   OUTC     OUTPUT PRINTABLE CHARACTER
0294 08             INX            POINT TO NEXT CHARACTER
0295 20 E4          BRA   PRINT1   FETCH CHAR AND PRINT
0297 08      PRINT4 INX            POINT TO LINE NUMBER
0298 DF 76          STX   SAVEX    SAVE IX
029A EE 00          LDX   0,X      SEE IF EOF
029C 27 04          BEQ   PRINT5   YES, THEN FINI
029E DE 76          LDX   SAVEX    RESTORE IX
02A0 20 C5          BRA   PRINT
02A2 7F 0040 PRINT5 CLR   NROUT
02A5 86 03          LDAA  #$03
02A7 BD 0023        JSR   OUTC
02AA 7E 0100        JMP   START
             *
02AD BD 0071 DELETE JSR   OUTSP    OUTPUT SPACE
02B0 BD 0329        JSR   INNR     INPUT NR
02B3 81 2C          CMPA  #',      SEE IF MULTIPLE DELETE
02B5 27 3A          BEQ   DELE5    YES
02B7 DE 32          LDX   STRMEM   GET START OF TEXT
02B9 DF 76   DELE1  STX   SAVEX    SAVE POINTER
02BB EE 00          LDX   0,X      LOAD LINE NR
02BD 27 2F          BEQ   DELE41   FLAG, NO SUCH LINE FOUND
02BF 9C 38          CPX   TEMPNR
02C1 27 10          BEQ   DELE4    FOUND LINE
02C3 DE 76          LDX   SAVEX    RETRIEVE POINTER
02C5 08             INX
02C6 8D 02          BSR   DELE2    FIND NEXT LINE
02C8 20 EF          BRA   DELE1
02CA 08      DELE2  INX
02CB A6 00          LDAA  0,X      GET CRITTER
02CD 81 8D          CMPA  #$8D     SEE IF CR
02CF 26 F9          BNE   DELE2    NO
02D1 08             INX            POINT TO NR
02D2 39             RTS
02D3 DE 76   DELE4  LDX   SAVEX    RETRIEVE IX
02D5 DF 2A          STX   NEW1
02D7 08             INX
        02D8 DELE41 EQU   *
02D8 8D F0          BSR   DELE2    FIND NEXT LINE
02DA DF 26          STX   BEGIN
02DC DE 34          LDX   ENDMEM   GET LAST OF FILE
02DE DF 28          STX   END      POINT TO END OF FILE
02E0 BD 01D9        JSR   BLOCK    PERFORM BLOCK MOVE
02E3 DE 2C          LDX   NEND
02E5 DF 34          STX   ENDMEM
02E7 6F 01          CLR   1,X
02E9 6F 02          CLR   2,X
02EB 7E 0100        JMP   START
02EE 7E 0119 DELE4F JMP   SKIP1
02F1 DE 38   DELE5  LDX   TEMPNR
02F3 DF 3C          STX   DEL1
02F5 BD 0329        JSR   INNR     GET NEXT LINE NUMBER TO DELETE
02F8 DE 38          LDX   TEMPNR
02FA DF 3E          STX   DEL2
02FC DE 32          LDX   STRMEM
02FE 96 3C          LDAA  DEL1
0300 D6 3D          LDAB  DEL1+1
0302 BD 0311        JSR   DELE6
0305 DF 2A          STX   NEW1
0307 96 3E          LDAA  DEL2
0309 D6 3F          LDAB  DEL2+1
030B BD 0311        JSR   DELE6
030E 7E 02D8        JMP   DELE41
             *
0311 BD 0309 DELE6  JSR   NIBCMP
0314 2F 12          BLE   DELE8
0316 36             PSHA
0317 BD 02CA        JSR   DELE2
031A 32             PULA
031B DF 76          STX   SAVEX
031D EE 00          LDX   0,X
031F 27 04          BEQ   DELE7
0321 DE 76          LDX   SAVEX
0323 20 EC          BRA   DELE6
0325 7E 0119 DELE7  JMP   SKIP1
0328 39      DELE8  RTS
             *
             *
0329 7F 0038 INNR   CLR   TEMPNR   CLR TEMP LOCATION
032C 7F 0039        CLR   TEMPNR+1
032F BD 0020 INNR1  JSR   GETC     GET DIGIT FROM TERMINAL
0332 84 7F          ANDA  #$7F     AND OFF HIGH ORDER BIT
0334 81 0D          CMPA  #$0D     SEE IF CR
0336 26 01          BNE   *+3      NO, BRANCH AROUND RETURN
0338 39             RTS
0339 81 2C          CMPA  #',      SEE IF DELIMITER
033B 26 01          BNE   *+3      NO, GO AROUND
033D 39             RTS
033E 81 03          CMPA  #$03     ALLOW FOR CANCEL COMMAND, CNTL-C
0340 26 03          BNE   INNR2    NO, GO ON
0342 7E 0100        JMP   START
0345 81 18   INNR2  CMPA  #$18     CHECK FOR CANCEL NR
0347 26 05          BNE   INNR3    NO
0349 BD 0071        JSR   OUTSP    OUTPUT SPACE
034C 20 DB          BRA   INNR     START OVER
```

```
0349 034E 81 2F   INNR3   CMPA   #$2F     SEE IF LESS THAN "0"
0350 0350 2A 03           BPL    INNR4
0351 0352 7E 0119         JMP    SKIP1    ERROR, OUTPUT "?" AND WAIT
0352 0355 81 39   INNR4   CMPA   #'9      SEE IF GT "9"
0353 0357 2E F9           BGT    INNR6-3  YES, ANOTHER ERROR
0354 0359 84 0F           ANDA   #$0F
0355 035B 36              PSHA            SAVE
0356 035C 96 38           LDAA   TEMPBR
0357 035E D6 39           LDAB   TEMPBR+1
0358 0360 58              ASLB
0359 0361 49              ROLA
0360 0362 97 38           STAA   TEMPBR   STORE 2*NUMBER
0361 0364 D7 39           STAB   TEMPBR+1
0362 0366 58              ASLB
0363 0367 49              ROLA
0364 0368 58              ASLB
0365 0369 49              ROLA
0366 036A 03 39           ADDB   TEMPBR+1 ADD IN TO GET 10*NUMBER
0367 036C D7 39           STAB   TEMPBR+1
0368 036E 00 38           ADCA   TEMPBR
0369 0370 97 38           STAA   TEMPBR
0370 0372 32              PULA            RETRIEVE DIGIT
0371 0373 9B 39           ADDA   TEMPBR+1
0372 0375 97 39           STAA   TEMPBR+1
0373 0377 20 03           BCC    INNR5
0374 0379 7C 0038         INC    TEMPBR
0375 037C 28 BE   INNR5   BRA    INNR1
                *
0376 037E DE 32   REMUN   LDX    STRMEM
0377 0380 7F 0030         CLR    LINEMR
0378 0383 7F 0031         CLR    LINEMR+1
0379 0386 BD 01CD REMUN1  JSR    INCLIN
0380 0389 DF 26           STX    SAVEX
0381 038B EE 00           LDX    0,X
0382 038D 27 09           BEQ    REMUN2
0383 038F BE 26           LDX    SAVEX
0384 0391 A7 00           STAA   0,X
0385 0393 E7 01           STAB   1,X
0386 0395 BD 02CA         JSR    DELE2
0387 0398 20 EC           BRA    REMUN1
0388 039A 97 3A   REMUN2  STAA   NEW1MB
0389 039C 97 3B           STAB   NEW1MB+1
0390 039E 7E 0100         JMP    START
                *
0391 03A1 DE 3A   COMP    LDX    NEW1MB
0392 03A3 DF 30           STX    LINEMB
0393 03A5 DE 34           LDX    ENDMEM
0394 03A7 96 3B           LDAA   LINEMB+1
0395 03A9 A7 00           STAA   0,X
0396 03AB 96 31           LDAA   LINEMB+1
0397 03AD A7 01           STAA   1,X
0398 03AF 7E 014E         JMP    NEW10
                *
0399 03B2 BD 0027 OUTSP   JSR    OUTCH
0400 03B5 96 36           LDAA   ENDMEM
0401 03B7 BD 07           BSR    LSR4A
0402 03B9 96 35           LDAA   ENDMEM+1
0403 03BB BD 07           BSR    LSR4A
0404 03BD BD 0F           BSR    OUTHR
0405 03BF 96 35           LDAA   ENDMEM+1
0406 03C1 BD 07           BSR    LSR4A
0407 03C3 BD 07           BSR    OUTHR
0408 03C5 7E 0100         JMP    START
0409 03C8 44      LSR4A   LSRA
0410 03C9 44              LSRA
0411 03CA 44              LSRA
0412 03CB 44              LSRA
                *
0413 03CC 84 0F   OUTHR   ANDA   #$0F
0414 03CE 8B 30           ADDA   #$30
0415 03D0 81 39           CMPA   #$39
0416 03D2 23 02           BLS    OUTHR1
0417 03D4 8A 07           ADDA   #$07
0418 03D6 7E 0023 OUTHR1  JMP    OUTCH
                *
0419 03D9 36      RTSCHP  PSHA
0420 03DA 37              PSHB
0421 03DB A0 00           SUBA   0,X
0422 03DD 27 03           BEQ    RTSCH
0423 03DF 33      RLT     PULB
0424 03E0 32              PULA
0425 03E1 39              RTS
0426 03E2 E0 01   RTSCH   SUBB   1,X
0427 03E4 27 F9           BEQ    RLT
0428 03E6 24 04           BCC    RTSCH1
0429 03E8 86 FF           LDAA   #$FF
0430 03EA 20 F3           BRA    RLT
0431 03EC 86 01   RTSCH1  LDAA   #$01
0432 03EE 20 EF           BRA    RLT
                *
                *
0433 03F0 7C 0040 SAVTPE  INC    NROW1
0434 03F3 BD 0020         JSR    GETC
0435 03F6 86 03           LDAA   #$03
0436 03F8 BD 0023         JSR    OUTC
0437 03FB DE 32           LDX    STRMEM
0438 03FD 7E 0379         JMP    PRINT0
                *
                *       LOAD TAPE
                *
0439 0400 7C 0040 LOTAPE  INC    NROW1
0440 0403 86 55           LDAA   #$55
0441 0405 B7 8004         STAA   $8004
0442 0408 BD 0020 LOTAP1  JSR    GETC
0443 040B 81 02           CMPA   #$02
0444 040D 26 F9           BNE    LOTAP1
0445 040F 7E 013E         JMP    NEW
                *
                *       INSERT NEW LINE
                *
0446 0412 BD 0071 INSERT  JSR    OUTSP
0447 0415 BD 0329         JSR    INNR       GET LINE NR
0448 0418 BD 0041         JSR    CRLF
0449 041B 96 3B           LDAA   TEMPBR+1
0450 041D B6 39           LDAB   TEMPBR+1
0451 041F BD 0220         JSR    CVBIN
0452 0422 BD 0071         JSR    OUTSP
0453 0425 DE 38           LDX    TEMPBR
0454 0427 86 02           LDAA   #$02     START COUNT AT 2
0455 0429 97 75           STAA   BLKCNT
0456 042B DF 81           STX    BUFFER   PUT LINE NUMBER IN BUFFER
0457 042D CE 0083         LDX    #BUFFER+2 POINT TO TEXT AREA
0458 0430 BD 0020 INSRT1  JSR    GETC     GET CHARACTER
0459 0433 81 03           CMPA   #$03     COMPARE FOR CANCEL
0460 0435 26 05           BNE    INSRT2
0461 0437 7E 0100         JMP    START    BAD CANCEL, GO BACK TO MON
0462 043A 81 08   INSRT2  CMPA   #$08     CHECK FOR BACKSPACE
0463 043C 2A 06           BPL    INSRT3
0464 043E 09              DEX             BACK UP ONE
0465 043F 7A 0075         DEC    BLKCNT
0466 0442 20 EC           BRA    INSRT1   GET NEXT CHARACTER
0467 0444 8A 80   INSRT3  ORAA   #$80     TURN ON HIGH ORDER BIT
0468 0446 7C 0075         INC    BLKCNT
0469 0449 A7 00           STAA   0,X      STORE CHARACTER
0470 044B 81 8D           CMPA   #$8D     SEE IF CR
0471 044D 27 03           BEQ    INSRT4
0472 044F 08              INX             MOVE POINTER TO NEXT LOCATION
0473 0450 20 DE           BRA    INSRT1   GET NEXT CHARACTER
                *
                *       BUFFER NOW HAS COMPLETE LINE
                *
0474 0452 DF 01   INSRT4  STX    BUFEND   SAVE POINTER TO LAST CHAR IN BUF
                *
                *       SEARCH FOR LINE TO MOVE
                *
0475 0454 DE 32           LDX    STRMEM
0476 0456 96 81           LDAA   BUFFER   GET LINE NUMBER
0477 0458 D6 82           LDAB   BUFFER+1
0478 045A BD 0389 INSRT5  JSR    RTSCMP
0479 045D 2B 07           BMI    INSRT6
0480 045F 26              PSHB
0481 0460 BD 02CA         JSR    DELE2
0482 0463 32              PULA
0483 0464 20 14           BRA    INSRT5
0484 0466 DF 26   INSRT6  STX    BEGIN
0485 0468 DE 34           LDX    ENDMEM
0486 046A DF 28           STX    END
0487 046C 9A 27           LDAA   BEGIN+1
0488 046E 90 75           SUBA   BLKCNT
0489 0470 97 29           STAA   NEW1+1
0490 0472 D6 00           LDAB   #$00
0491 0474 99 26           ADCA   BEGIN
0492 0476 97 28           STAA   NEW1
0493 0478 00 0109         JSR    BLOCK
0494 047B DE 2C           LDX    NEW
0495 047D DF 24           STX    ENDMEM
0496 047F 6F 00           CLR    0,X
0497 0481 6F 01           CLR    1,X
0498 0483 DE 26           LDX    BEGIN
0499 0485 DF 2A           STX    NEW1
0500 0487 CE 0081         LDX    #BUFFER
0501 048A DF 2C           STX    BEGIN
0502 048C DE 01           LDX    #BUFEND
0503 048E 6F 28           CLR    END
0504 0490 BD 0109         JSR    BLOCK
0505 0493 7E 0100         JMP    START
0506                      END

SYMBOL TABLE

INEEE  E1AC   OUTEEE E101   PDATA1 E07E   ACIA1  8004
ACIAD  8005   GETC   0020   OUTC   0023   BEGIN  0026
END    0028   NEW    0024   NEW    002C   SAVE   002E
LINEMB 0030   STRMEM 0032   ENDMEM 0034   BACKCK 0036
TEMPBR 0038   NEXTWD 003A   DEL1   003C   DEL2   003E
NBUF   0040   CBLF   0081   CAN1   0058   CAN2   005B
BL11   006A   PAMP1  00A5   OUTSP  0071   BLKCNT 0075
SAVEX  0076   SAVEA  0078   H10K   0079   BUFFER 0081
BUFEND 00D1   START  0100   LBOX   0109   SKIP   0111
SKIP1  0119   BTABLE 0120   TABLE0 013E   MEM    013E
MEM10  014E   MEW11  015F   INCH   0181   INCMD  0181
INCMP  0187   INCH2  0190   INCH3  019C   INCH4  01A5
ERROR  01AE   INCLIN 01CD   INCLE1 01D4   BLOCK  01DF
BLOCK1 01E7   TEST1  01E7   TCDT2  0177   FRONT  01FD
LOOP1  0202   BACK   0201   LOOP2  0214   CVBIN  0220
CVDEC1 0225   CVDEC2 0228   CVDEC5 0233   GETLN0 024A
PRINTS 024F   PRINT6 0258   PRINT  0267   PRINT0 0270
PRINT1 0278   PRINT2 0282   PRINT3 0288   PRINT4 0297
PRINT5 02A2   DELETE 02AB   DELE1  02B9   DELE2  02CA
DELE4  02D3   DELE41 02DB   DELE41 02EE   DELE5  02F1
DELE6  0311   DELE7  0325   DELE8  0328   INNR   0329
INNR1  0329   INNR2  0345   INNR3  034E   INNR4  0355
INNR5  037C   RENUM  037E   RENUM1 0386   RENUM3 039A
CONT   03A1   QUERY  03B2   LBR4A  03C8   OUTHR  03DC
OUTH1  03D6   RTSCMP 03D9   RLT    03DF   RTSEQ  03E2
RTS11  03EC   SAVTPE 03F0   LOTAPE 0400   LOTAP1 0408
INSERT 0412   INSRT1 0430   INSRT2 043A   INSRT3 0444
INSRT4 0453   INSRT5 045A   INSRT6 0466

  0 ERROR(S)
```

```
*J 0100
>H
0010
0020 ************************************
0030
0040      THIS IS A NEW LINE EDITOR
0050
0060 IT IS A SIMPLE LINE ORIENTED EDITOR WITH SOME BASIC, BUT POWERFUL
0070 COMMANDS.  EACH LINE IS STORED IN MEMORY PRECEDED BY A TWO BYTE
0080 BINARY UNSIGNED INTEGER AND TERMINATED WITH A CARRIAGE RETURN (CR).
0090
0100 THE EDITOR COMMAND CONSIST OF ONE LETTER COMMANDS AND AN OPTIONAL
0110 PARAMETER OR PARAMETERS AS ILLUSTRATED BELOW.  THIS EDITOR IS
0120 WRITTEN FOR CREATING ASSEMBLY LANGUAGE PROGRAMS TO BE ASSEMBLED
0130 BY AN ACCOMPANYING ASSEMBLER, BUT MAY BE USED FOR THE CREATION OF
0140 TEXT NOTREQUIRING JUSTIFICATION COMMANDS, OR THE READER MAY ADD
0150 SUCH COMMANDS EASILY, SINCE THE SYSTEM IS PROCEDURE ORIENTED.
0160
0170 THE COMMANDS ARE AS FOLLOWS:
0180
0190     P  (CR)      PRINT FROM BEGINNING.  PRINT MAY BE TERMINATED BY CNTL-C.
0200
0210     DPP N                          <-- ERROR IN TYPING, TO BE DELETED.
0220     P  N         PRINT FROM LINE NUMBER N.  ANY PRINT MAY BE CANCELED BY CNTL-C.
0230
0240     D  N         DELETE LINE NUMBERED N
0250
0260     D  N,M       DELETE LINE NUMBERS N THRU M INCLUSIVE
```

NOTE: TYPOGRAPHICAL ERROR CORRECTED BY EDITOR.

```
0270
0280     ?          OUTPUT LAST MEMORY LOCATION OCCUPIED BY TEXT
0290
0300     I  N       INSERT LINE NUMBER N
0310
0320     S          SAVE SOURCE TO TAPE.  STARTS WHEN KEY IS TYPED AFTER S.
0330
0340     L          LOAD SOURCE FROM TAPE.
0350
0360     CNTL-C     END OF TEXT ENTRY MODE OR CANCEL PRINT AFTER CR/LF OF NEXT LINE.
0370
0380     N          START NEW TEXT IN MEMORY, WILL DELETE OLD TEXT.
0390
0400     E          EXIT EDITOR AND RETURN TO MONITOR
0410
0420     ②— CNTL-C TYPED HERE.
>

 P
0010
0020 ********************************************
0030
0040      THIS IS A NEW LINE EDITOR
0050
0060 IT IS A SIMPLE LINE ORIENTED EDITOR WITH SOME BASIC, BUT POWERFUL
0070 COMMANDS.  EACH LINE IS STORED IN MEMORY PRECEDED BY A TWO BYTE
0080 BINARY UNSIGNED INTEGER AND TERMINATED WITH A CARRIAGE RETURN (CR).
0090
0100 THE EDITOR COMMANDS CONSIST OF ONE LETTER COMMANDS AND AN OPTIONAL
0110 PARAMETER OR PARAMETERS AS ILLUSTRATED BELOW.  THIS EDITOR IS
0120 WRITTEN FOR CREATING ASSEMBLY LANGUAGE PROGRAMS TO BE ASSEMBLED
0130 BY AN ACCOMPANYING ASSEMBLER, BUT MAY BE USED FOR THE CREATION OF
0140 TEXT NOT REQUIRING JUSTIFICATION COMMANDS, OR THE READER MAY ADD
0150 SUCH COMMANDS EASILY, SINCE THE SYSTEM IS PROCEDURE ORIENTED.
0160
0170 THE COMMANDS ARE AS FOLLOWS:
0180
0190     P  (CR)    PRINT FROM BEGINNING.  PRINT MAY BE TERMINATED BY CNTL-C.
0200
0210 P
0220     P  N       PRINT FROM LINE NUMBER N.  AGAIN PRINT MAY BE CANCELED BY CNTL-C.
0230
0240     D  N       DELETE LINE NUMBERED N
0250
0260     D  N,M     DELETE LINE NUMBERS N THRU M INCLUSIVE
0270
0280     ?          OUTPUT LAST MEMORY LOCATION OCCUPIED BY TEXT
0290
0300     I  N       INSERT LINE NUMBER N
0310
0320     S          SAVE SOURCE TO TAPE.  STARTS WHEN KEY IS TYPED AFTER S.
0330
0340     L          LOAD SOURCE FROM TAPE.
0350
0360     CNTL-C     END OF TEXT ENTRY MODE OR CANCEL PRINT AFTER CR/LF OF NEXT LINE.
0370
0380     N          START NEW TEXT IN MEMORY, WILL DELETE OLD TEXT.
0390
0400     E          EXIT EDITOR AND RETURN TO MONITOR
0410
>C
0420     N          RENUMBER LINES IN MEMORY.
0430
0440     C          CONTINUE ADDING TO FILE.
0450
0460     ③
>
```

```
 P
0010
0020 ********************************
0030
0040      THIS IS A NEW LINE EDITOR
0050
0060 IT IS A SIMPLE LINE ORIENTED EDITOR WITH SOME BASIC, BUT POWERFUL
0070 COMMANDS.  EACH LINE IS STORED IN MEMORY PRECEDED BY A TWO BYTE
0080 BINARY UNSIGNED INTEGER AND TERMINATED WITH A CARRIAGE RETURN (CR).
0090
0100 THE EDITOR COMMANDS CONSIST OF ONE LETTER COMMANDS AND AN OPTION L
0110 PARAMETER OR PARAMETERS AS ILLUSTRATED BELOW.  THIS EDITOR IS
0120 WRITTEN FOR CREATING ASSEMBLY LANGUAGE PROGRAMS TO BE ASSEMBLED
0130 BY AN ACCOMPANYING ASSEMBLER, BUT MAY BE USED FOR THE CREATION OF
0140 TEXT NOT REQUIRING JUSTIFICATION COMMANDS, OR THE READER MAY ADD
0150 SUCH COMMANDS EASILY, SINCE THE SYSTEM IS PROCEDURE ORIENTED.
0160
0170 THE COMMANDS ARE AS FOLLOWS:
0180
0190     P  (CR)    PRINT FROM BEGINNING.  PRINT MAY BE TERMINATED BY CNTL-C.
0200
0210 P
0220     P  N       PRINT FROM LINE NUMBER N.  AGAIN PRINT MAY BE CANCELED BY CNTL-C.
0230
0240     D  N       DELETE LINE NUMBERED N
0250
0260     D  N,M     DELETE LINE NUMBERS N THRU M INCLUSIVE
0270
0280     ?          OUTPUT LAST MEMORY LOCATION OCCUPIED BY TEXT
0290
0300     I  N       INSERT LINE NUMBER N
0310
0320     S          SAVE SOURCE TO TAPE.  STARTS WHEN KEY IS TYPED AFTER S.
0330
0340     L          LOAD SOURCE FROM TAPE.
0350
0360     CNTL-C     END OF TEXT ENTRY MODE OR CANCEL PRINT AFTER CR/LF OF NEXT LINE.
0370
0380     N          START NEW TEXT IN MEMORY, WILL DELETE OLD TEXT.
0390
0400     E          EXIT EDITOR AND RETURN TO MONITOR
0410
0420     R          RENUMBER LINES IN MEMORY.
0430
0440     C          CONTINUE ADDING TO FILE.
0450
>D 210      DELETE 21⌀
>P 160
0160
```

```
0170 THE COMMANDS ARE AS FOLLOWS:
0180
0190     P  (CR)    PRINT FROM BEGINNING.  PRINT MAY BE TERMINATED BY CNTL-C.
0200
0210     P  N       PRINT FROM LINE NUMBER N.  AGAIN PRINT MAY BE CANCELED BY CNTL-C.
0220
0230     D  N       DELETE LINE NUMBERED N
0240
0250

>P
0010
0020 ********************************
0030
0040      THIS IS A NEW LINE EDITOR
0050
0060 IT IS A SIMPLE LINE ORIENTED EDITOR WITH SOME BASIC, BUT POWERFUL
0070 COMMANDS.  EACH LINE IS STORED IN MEMORY PRECEDED BY A TWO BYTE
0080 BINARY UNSIGNED INTEGER AND TERMINATED WITH A CARRIAGE RETURN (CR).
0090
0100 THE EDITOR COMMANDS CONSIST OF ONE LETTER COMMANDS AND AN OPTIONAL
0110 PARAMETER OR PARAMETERS AS ILLUSTRATED BELOW.  THIS EDITOR IS
0120 WRITTEN FOR CREATING ASSEMBLY LANGUAGE PROGRAMS TO BE ASSEMBLED
0130 BY AN ACCOMPANYING ASSEMBLER, BUT MAY BE USED FOR THE CREATION OF
0140 TEXT NOT REQUIRING JUSTIFICATION COMMANDS, OR THE READER MAY ADD
0150 SUCH COMMANDS EASILY, SINCE THE SYSTEM IS PROCEDURE ORIENTED.
0160
0170 THE COMMANDS ARE AS FOLLOWS:
0180
0190     P  (CR)    PRINT FROM BEGINNING.  PRINT MAY BE TERMINATED BY CNTL-C.
0200
0220     P  N       PRINT FROM LINE NUMBER N.  AGAIN PRINT MAY BE CANCELED BY CNTL-C.

>D 10,50      DELETE RANGE 1⌀ → 5⌀
>P
0060 IT IS A SIMPLE LINE ORIENTED EDITOR WITH SOME BASIC, BUT POWERFUL
0070 COMMANDS.  EACH LINE IS STORED IN MEMORY PRECEDED BY A TWO BYTE
0080 BINARY UNSIGNED INTEGER AND TERMINATED WITH A CARRIAGE RETURN (CR).
0090
0100 THE EDITOR COMMANDS CONSIST OF ONE LETTER COMMANDS AND AN OPTIONAL
0110 PARAMETER OR PARAMETERS AS ILLUSTRATED BELOW.  THIS EDITOR IS
0120 WRITTEN FOR CREATING ASSEMBLY LANGUAGE PROGRAMS TO BE ASSEMBLED
0130 BY AN ACCOMPANYING ASSEMBLER, BUT MAY BE USED FOR THE CREATION OF
0140 TEXT NOT REQUIRING JUSTIFICATION COMMANDS, OR THE READER MAY ADD
0150 SUCH COMMANDS EASILY, SINCE THE SYSTEM IS PROCEDURE ORIENTED.
0160
0170 THE COMMANDS ARE AS FOLLOWS:
0180
0190     P  (CR)    PRINT FROM BEGINNING.  PRINT MAY BE TERMINATED BY CNTL-C.
 300
 270     P  N       PRINT FROM LINE NUMBER N.  AGAIN PRINT MAY BE CANCELED BY CNTL-C.

>R      RENUMBER
>P
0010 IT IS A SIMPLE LINE ORIENTED EDITOR WITH SOME BASIC, BUT POW  FUL
0020 COMMANDS.  EACH LINE IS STORED IN MEMORY PRECEDED BY A TWO BYTE
0030 BINARY UNSIGNED INTEGER AND TERMINATED WITH a CARRIAGE RETURN (CR).
0040
0050 THE EDITOR COMMANDS CONSIST OF ONE LETTER CO  AND AND AN OPTIONAL
0060 PARAMETER OR PARAMETERS AS ILLUSTRATED BELOW.  THES EDITOR IS
0070 WRITTEN FOR CREATING ASSEMBLY LANGUAGE PROGRAMS TO BE ASSEMBLED
0080 BY AN ACCOMPANYING ASSEMBLER, BUT MAY BE USED FOR THE CREATION OF
0090 TEXT NOT REQUIRING JUSTIFICATION COMMANDS, OR THE READER MAY ADD
0100 SUCH COMMANDS EASILY, SINCE THE SYSTEM IS PROCEDURE ORIENTED.
```

FLEX QUICKIE ROUTINES

James Caldwell
Box 1601
Port Isabel, TX 78578

PAUSE CONTROL

Did you ever use the print command and then go back to your vidio terminal and find that the pause feature of 'FLEX' no longer functioned, or if you use the PORT command that Mickey Ferguson will probably publish in a future issue of the '68' Micro Jornal to direct all I/O to another port, like my TTY-43 on port No. 3, and find that the pause that is so great on a CT-1024 ain't worth a durn on the tty.

Well here is a way out, for us folks that still have a paging vidio terminal on port No. 1, actualy here are three little utilities that I use on my system.

H.CMD will home and clear a CT-1024 type terminal and can be used as a multiple comand IE H:cat or H:list H.TXT

PC.CMD will home and clear the 1024 and reset the pause feature for the vidio terminal.

NP.CMD will defeat the pause feature if you want to look at a long listing on the vidio terminal or if you are using the TTY as I/O with Mickeys PORT command. and dont want to have to poke the ESC key every few lines

James D. Caldwell
Box 1601
Port Isabel tx 78578

```
                          NAM    FLEX H.CMD
                    * HOME AND CLEAR   *
                    * FOR THE CT-1024  *
                    *  JIM CALDWELL    *
                    * OCTOBER 8 1978   *

7118                PSTRNG EQU    $7118
7103                WARMS  EQU    $7103
7600                       ORG    $7600
7600 20 01          CLEAR  BRA    CLEAR2
7602 01             VN     FCB    1
7603 CE 76 09       CLEAR2 LDX    #STRNG
7606 BD 71 18              JSR    PSTRNG
7609 10             STRNG  FCB    16,22,04
760A 16 04
760C 7E 71 03              JMP    WARMS
                           END    CLEAR

NO ERROR(S) DETECTED

                           NAM    PC.CMD

                    **********************
                    * RESETS THE PAUSE *
                    * FEATURE IN FLEX  *
                    * AND H&C THE 1024 *
                    **********************

7089                REPAUSE EQU    $7089
00FF                VALUE  EQU    $FF
7103                WARMS  EQU    $7103
7118                PSTRNG EQU    $7118
7600                       ORG    $7600
7600 20 01          R      BRA    RP
7602 01             VN     FCB    1
7603 86 FF          RP     LDA A  #VALUE
7605 B7 70 89              STA A  REPAUSE
7608 10             HC     FCB    $10,$16,$04
7609 16 04
760B CE 76 08              LDX    #HC
760E BD 71 18              JSR    PSTRNG
7611 7E 71 03              JMP    WARMS
                           END    R

NO ERROR(S) DETECTED
```

```
                           NAM    NP.CMD

                    **********************
                    * SETS THE PAUSE *
                    * FEATURE IN FLEX  *
                    **********************

7089                REPAUSE EQU    $7089
0000                VALUE  EQU    $00
7103                WARMS  EQU    $7103
7600                       ORG    $7600
7600 20 01          R      BRA    RP
7602 01             VN     FCB    1
7603 86 00          RP     LDA A  #VALUE
7605 B7 70 89              STA A  REPAUSE
7608 7E 71 03              JMP    WARMS
                           END    R

NO ERROR(S) DETECTED
```

NEW PRODUCTS

Several new software products have been announced by Star-Kits, P. O. Box 209, Mt. Kisco NY 10549. They all run on SWTP 6800-based computers, using either SWTP MF-68 or Percom LFD-400 disk systems, and all include full source code. Six software packages have been announced:

A FULL-DISK SORT-MERGE written in Basic, which can sort complicated files, as large as a full disk, in various ways. Price is $35 on disk, $30 with listing only.

BASIC UTILITY PACKAGE renumbers Basic programs, does pretty-printing automatically, reduces memory requirements, prints an index of all variables used and their locations, prints table of all program transfers, helps keep track of program modifications, and more. Price on disk is $30.

INTERRUPT-DRIVEN TERMINAL DRIVER interfaces the main terminal or a serial printer to Basic or machine language programs in such a way that both can run at the same time. Price is $20 on cassette, $25 on Percom disk.

CHECKBOOK BALANCING PACKAGE does a complete balancing of a checkbook, keeps track of outstanding checks or deposits, maintains a year-to-date file, provides income tax summaries, and more. Especially good for two husband-wife checkbooks on one account. Price is $40.

ELIZA in both Basic and machine language. Doesn't require disk. Price is $15 on disk or cassette, only $5 when combined with another purchase on same disk or cassette.

FLOGEN automatically flowcharts Basic programs. Helps find those subtle errors; great for long-term documentation. Requires 72-column printer. Free with any purchase.

BASIC SPEED UPDATES

Keith Alexander
681 Whitmore Rd., No. 207
Detroit, MI 48203
SEPT. 9, 1979

TEL: 313-861-6137

'68 MICRO JOURNAL
3018 NAMILL RD.,
P.O.BOX 849
NIXON, TN

DEAR SIRS

HAVING READ OF TSC'S (TECHNICAL SYSTEMS CONSULTANTS) CLAIMS FOR THEIR BASICS (6800 AND '09), AND HAVING USED BOTH OF THEM, I THOUGHT YOUR READERS MIGHT BE INTERESTED IN THE TIME TRIAL RESULTS ATTACHED.

THESE NEW BASICS OBVIOUSLY CAN OUTRUN A Z-80 AT FULL CHAT (4 MHZ), BUT IT WOULD SEEM THAT THE OSI MACHINE (6502) IS STILL A LITTLE FASTER, WITH QUALIFICATIONS. TO WIT, THE OSI SYSTEM'S CPU AND MEMORY ARE GUARANTEED TO RUN AT 2 MHZ, BUT NOT THE DISK CONTROLLER.
THIS WAS EXPLAINED IN THE ORIGINAL OCT.'77 KILOBAUD ARTICLE. THEREFORE, THE FASTEST TIMES REPORTED WERE FOR A NON-DISK SYSTEM. SIMILARLY, YOU COULD PROBABLY GET EVEN BETTER TIMES WITH AN INTEGER-ONLY BASIC, BUT WHO WANTS SUCH COMPROMISES? PERSONALLY, I WANT MY SYSTEM SOFTWARE TO BE TOTALLY DISK-INTERACTIVE AND HAVE FLOATING-POINT CAPABILITY. SPEED ISN'T OF PRIMARY IMPORTANCE, (UNLESS YOU'RE PLAYING A CODEBREAKER/MAKER GAME!), BUT IT'S NICE TO KNOW I'VE GOT THE FASTEST GUN AROUND.

VERY TRULY YOURS,

KEITH ALEXANDER

IT'S BEEN TWO YEARS SINCE TOM RUGG AND PHIL FELDMAN'S ARTICLES WERE PUBLISHED IN KILOBAUD, ON THE SPEED COMPARISONS OF VARIOUS MICROCOMPUTER BASIC INTERPRETERS. THE SEVEN BENCHMARK PROGRAMS THEY USED FOR THEIR TESTS ARE DEMONSTRATIVE OF THE SIMPLER FUNCTIONS FOUND IN ANY BASIC: SIMPLE ARITHMETIC, THE IF-THEN CONSTRUCT, ITERATIVE(LOOPING) OPERATIONS, AND ONE-DIMENSIONAL ARRAY MANIPULATION. I THINK THEY'VE JUSTIFIABLY BECOME SOMEWHAT OF A STANDARD FOR COMPARISON OF MICRO BASICS.

FOR THE SAKE OF SPACE, I'LL REFER YOU TO THE JUNE AND OCTOBER '77 ISSUES OF KILOBAUD FOR THE ACTUAL SOURCE LISTINGS. FROM HERE ON I'LL REFER TO THEM SIMPLY AS BENCHMARKS 1 THROUGH 7.

I'M AFRAID THE SWTPC BASIC WAS DISMALLY SLOW COMPARED TO MANY OTHERS IN THE ORIGINAL TESTS. NOW, DON'T GET ME WRONG, SWTP'S BASIC WAS FINE WITH ME. NOT MANY OTHER MICRO BASICS HAVE NINE DIGITS OF FLOATING POINT PRECISION, AND IF IT'S SPEED I WANT, THERE'S ALWAYS THE "USER" FUNCTION AND ASSEMBLY LANGUAGE. LET'S FACE IT, BASIC IS A LANGUAGE OF CONVENIENCE AND EASE, NOT NECESSARILY BLINDING SPEED.

AND NOW FOR THE BIG "BUT."

BUT THEN TSC (TECHNICAL SYSTEMS CONSULTANTS) BROUGHT OUT THEIR BASIC FOR THE 6800, OPERATING UNDER THEIR FLEX DISK OPERATING SYSTEM(DOS), AND PROMPTLY CLAIMED IT WAS THE FASTEST BASIC AVAILABLE FOR MICROS. WELL, THE FIGURES ARE OUT.

I MUST STRESS THAT THE TIMES GIVEN FOR MY TESTS ON THE FLEX BASICS WERE TAKEN WITH THE SECOND HAND OF MY ROLEX, NOT A STOPWATCH. ALSO, TO AID IN HELPING THIS TIMING METHOD, I INSERTED A "PRINT CHR$(07)" (BELL) AT THE END OF EACH BENCHMARK.

THE TIMES GIVEN ARE FOR SEVERAL PROCESSORS:
COLUMN "A" IS THE ORIGINAL SWTPC 8K BASIC 1.0.
COLUMN "B" IS FOR ZAPPLE 8K BASIC 1.1 IN AN ALTAIR 8800 WITH A CROMEMCO Z-80 AT 4 MHZ WITH ONE WAIT STATE.
COLUMN "C" IS THE PREVIOUS WINNER, AN OSI 8K BASIC (VER 1.0, REV 3.3-6502 CPU) RUNNING AT 2.0 MHZ. MORE ON THIS LATER.
COLUMN "D" IS TSC BASIC IN MY 1.0 MHZ SYSTEM, AND COLUMN "E" IS TSC 6809 BASIC IN THE SAME MACHINE.

PROGRAM	SYSTEM/INTERPRETER				
	A	B	C	D	E
BENCHMARK #1	14.9	0.9	0.9	--	--
BENCHMARK #2	24.7	5.9	4.6	4.5	3.5
BENCHMARK #3	96.1	13.0	8.2	11.5	11.0
BENCHMARK #4	105.3	13.5	9.3	11	11.3
BENCHMARK #5	109.8	14.8	10.0	11.8	12
BENCHMARK #6	174.1	22.7	14.0	18	17.5
BENCHMARK #7	204.5	32.7	21.6	29	27

(ALL TIMES IN SECONDS)

BENCHMARK #1 EXECUTED TOO FAST TO BE TIMED BY THIS METHOD (LESS THAN 1 SECOND).

COLUMN "C", REPRESENTING THE OSI CHALLENGER TIMES, IS BASED ON A 2-MHZ SYSTEM CLOCK, WHICH IS ATTAINABLE ONLY IN A NON-DISK SYSTEM. SEE THE OCT.77 KILOBAUD P.21.

COLUMN "B", THE CROMENCO 4-MHZ Z-80, REPRESENTED THE FASTEST OF THAT SORRY BREED.

NOW, DOES ANYONE HAVE A 2-MHZ 6800 OR '09 SYSTEM OPERATING UNDER FLEX 2.0 OR 9.0 ? LET'S INTRODUCE THESE PEOPLE TO A NEW CONCEPT OF SPEED !

FLEX(tm) Users Group

A letter received from:
Richard Cagle
11103 Sagepark Ln
Houston, TX 77089

Suggest I acknowledge the FLEX Users Group. So here it is Richard, sorry I can not go into more detail but it seems I know very little about it, maybe they or you will send me more info. Am always glad to plug anything that helps 6800/09 users.

FLEX Users Group
% Ron Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

If any of the rest of you know of publications and/or organizations that we as 6800/09 users would be interested in, please let me know so I can pass the word along!

P.S.
Richard for black PR-40 ribbons try:

Addressograph Corp
Part number: Series 7000
LRC Printer Ribbon - Black
Reorder no.:116-2395-896
A-M Graphics Nylon

Check phone book for local number.

THE TERMINAL

Great Plains Computer Company, Inc. has developed a terminal package for FLEX 2 on the TANO OUTPOST 11. This powerful program package allows the computer to act as an on-line terminal to host computers, or as the host computer itself. It also provides for transfer of disk files to and from the host. The program package is available on mini-disk with an easily understood manual for $150.00 from GREAT PLAINS COMPUTER COMPANY, INC., P.O. BOX 916, IDAHO FALLS, IDAHO 83401. (208) 529-3210

RENUMBER PATCH (PERCOM)

The following is a patch to the BASIC Renumber (June 79, 68 Micro Journal) article, by Mickey Ferguson, for use with PERCOM Super BASIC.

RENUM6 routine is changed as follows:

```
  CMPA 2
  BNE RENU65
  CMPB $6F
  BEQ CO
  CMPB $68
  BEQ CO

NEXTBA EQU $28
SOURCE EQU $2C
MSLINE EQU $30
LSLINE EQU $31
```

Submitted by:

Ken McCullough
8400 Broadway #157
Houston, TX 77061

Thanks Ken for the offering. I and a lot of other 6800/09 users appreciate your input. Have extended your subscription for this gem.

O.K. you other users, get your subscription extended, just send it to 68 Micro Journal, article that is. Remember if we don't share we defeat the purpose of this endevor. We are all in this thing together. By sharing our expertise we all gain. If you have a routine or program, send it in (ready to run if possible), typed and spaced for our page size. We extend your subscription and a lot of users thank you.

PROCEDURE LIBARARY

Don Williams
'68' MICRO JOURNAL
3018 Hewitt Rd.,
Hilton, Tennessee 37343

Gary Magnusen
208 Tinkler
Lafayette, IN 47901

I HAD AN IDEA ON SUNDAY

After keying and re-keying COMMONLY used functions, such as pursing diskettes, copying files, editing-assembling-testing, ETC, using the same programs, and data files, I decided to develope a 'PROCEDURE-LIBRARY' to handle this repetitive and often mis-keyed task for me.

The primary objective was to key those functions ONE TIME ONLY, and have this program INTERACT with the Disk Operating System.

With this in mind, I wrote Two Programs. One to fetch an existing Procedure OR create the Procedure, then establish ADDRESSIBILITY with the interactive phase. The second, to extract and move these pre-keyed lines to the DOS line buffer, where they could be acted upon, as if they were just keyed.

The SECOND program can be called by any phase in execution, to extract file data, test data, etc.

If the documentation or Source programs are to lengthly to Print or Key, I would be happy to give copies to anyone who will send me a 5" diskette or Pay the cost ($4.25) of one, and one dollar for postage, packaging, printing, etc.

Thank you

Gary M. Magnusen
208 Tinkler
Lafayette, Indiana 47901
(I-317-742-1565)

P.S.
Would you please send me your advertising price schedule. I have a program that uses Verbs such as READ, WRITE, MOVE ACCEPT, COMPARE, etc and generates 6800 code, ready to assemble. I think its worth about $10.00. I'm also writing an ANS COBOL Compiler, about $50.00 for it.

PROGRAM NUMBER 1:

This Program loads into the DOS T.C.A., and is called to extract an existing procedure, or create one, if no file name is given.

Two types of items may be entered:
A. COMMAND LINES:
Any Key-Word with attributes to be interpreted by DOS.

B. DATA LINES:
Any items to be accepted by your calling program.

The Command Lines are preceded by the Control character (02), seen only on disk.

This program will display a title showing which type of item is being entered, and each line will be prefixed with a PERIOD to indicate a COMMAND LINE or a SPACE to indicate a DATA LINE.

To switch from Command Line to Data Line and vice-versa, HIT ESCAPE then a C/R at the beginning of a line. This can be done as many times as desired.

When completed, to exit this program, Hit C/R at the beginning of a line.

PROGRAM NUMBER 2:

This program resides with the DISK OPERATING SYSTEM and must be loaded with it when you bring the system up.

This program has Two entry points.

FROM DOS:
This entry will read on a line by line basis and place that line into the DOS line buffer. Then control is returned to DOS. Only COMMAND LINES will be entered. All DATA LINES will be displayed and ignored.

FROM ANY EXECUTING PROGRAM:
This entry will read on a line by line basis and place that line into the DOS line buffer. Then control is returned to the calling program.
The DOS command flag is lowered to indicate the type of line passed. (SET=Command Line/CLEAR=Data Line)
It's the users responsibility to test this flag.

I've found it easy to use a END-OF-DATA record with any data file that has an unknown number of records.

DOS MODIFICATIONS:
I have added Two user Commands which are on the front of program number 2.

A. GO NNNN:
Where N is any valid HEX address.
This allows the using of GET commands then GO commands, bypassing the need to SAVE the program just to establish a transfer address.

B. PAUSE:
The system will wait for a new Command Line, then act upon it, or if C/R is entered, will continue processing.

This allows you to swap diskettes, go to your monitor, etc, before continuing.

```
PROCA    COMMAND/DATA PROCEDURE          6SS 6NEMONIC ASSEMBLER    PAGE 1

                  NAM     PROC.A        COMMAND/DATA PROCEDURES
                  OPT     PAG:NOS
A700              ORG     $A700

          *  SYSTEM.....SWTP/DOS68.42A
          *  DATE WRITTEN.....AUG 3, 1979
          *  AUTHOR.....GARY A. MAGNUSEN

          ************************************************
          *       THIS PROGRAM RESIDES WITHIN DOS
          ************************************************


          ************************************************
          *       TWO USER COMMANDS HAVE BEEN ADDED; SEE BELOW.
          ************************************************
```

```
A700            UCTBL   EQU     *               USER COMMAND TABLE
A700 02                 FCD     2,2
A701 02
A702 A7 10              FDB     GOTO
A704 47                 FCC     'GO'
A705 4F

A706 05                 FCB     5,5
A707 05
A708 A7  0              FDB     PAUSE
A70A 50                 FCC     'PAUSE'
A70B 41 55
A70D 53 45

A70F 00                 FCB     0               END OF USER TABLE

A710            GOT     EQU     *               GO TO (ADDRESS)
A710 BD B2 A0           JSR     ZGETHN          GET HEX ADDRESS
A713 24 06              BCC     GOTO10          BRANCH IF GOOD ADDRESS
A715 CE B6 AE           LDX     #INVAL          GET INVALID MESSAGE
A718 7E B2 9D           JMP     ZDIE            DISPLAY & RETURN TO DOS

A71B 8E A0 42   GOTO10  LDS     #STACK          USE SWTBUG STACK
A71E 6E 00              JMP     0,X             GO TO ADDRESS WANTED

A720            PAUSE   EQU     *               PAUSE
A720 BD B2 B5           JSR     ZLINEI          EXCEPT NEW LINE
A723 FE B2 FC           LDX     LINPTR          CHECK
A726 A6 01              LDA A   1,X             FIRST CHARACTER
A728 81 0D              CMP A   #$0D            TO C/R
```

PR A COMMAND/DATA PROCEDURE SSB MNEMONIC ASSEMBLER PAGE 2

```
A72A 27 2E              BEQ     PROC90          IF SO, RETURN THRU RTS
A72C 7E B2 E8           JMP     $B2E8           IF NOT, GO ACT UPON COMMAND LINE
```

```
*
*****************************************************
*       THIS SECTION WILL READ ONE LINE OF INFO FROM DISK A
*       PLACE IT INTO THE DOS LINE BUFFER.
*
*       IF ENTRY IS THRU $A72F, ONLY COMMAND LINES WILL BE
*       ACCEPTED. ALL OTHERS WILL BE BYPASSED.
*
*       IF ENTRY IS THRU $A73B, ALL LINES WILL BE ACCEPTED
*       AND THE DOS COMMAND FLAG WILL BE SET.
*****************************************************
```

```
A72F            JCPROC  EQU     *               READ PROCEDURE-LIBRARY
A72F 7D A7 FF           TST     PROCFG          TEST PROC FLAG
A732 26 1A              BNE     PROC10          IF SET, GO READ DISK
A734 7D A7 FA           TST     PROCF2          TEST ALTERNATE PROC FLAG
A737 26 03              BNE     PROC            IF SO
A739 7E B2 B5           JMP     ZLINEI          RETURN TO DOS THRU GET DOS LINE RTN

A73C FE A7 F6   PROC    LDX     SSUNI2          SWAP OUT ALTERNATE/PRIMARY PROCS
A73F FF A7 FB           STX     SSUNIT
A742 FE A7 F8           LDX     SSNPO2
A745 FF A7 FD           STX     SSNPOS
A748 7F A7 FA           CLR     PROCF2
A74B 7C A7 FF           INC     PROCFG

A74E 8D 0B      PROC10  BSR     SSREAD          READ ONE LINE
A750 7D A7 FF           TST     PROCFG          IS ANY LINE PRESENT?
A753 27 DA              BEQ     JCPROC          IF NOT, TEST ALTERNATE PROC
A755 7D A7 F5           TST     DOSFLG          IS COMMAND LINE PRESENT?
A758 27 F4              BEQ     PROC10          IF NOT, GO READ NEXT LINE
A75A 39         PROC90  RTS                     RETURN TO DOS AFTER GET DOS LINE
```

```
                ***** SINGLE SECTOR READ *****

A75B            SSREAD  EQU     *
A75B 7F A7 F5           CLR     DOSFLG          CLEAR DOS COMMAND LINE FLAG
A75E CE AF FF           LDX     #LB-1           GET DOS LINE BUFFER ADDRESS
A761 FF B2 FC           STX     LINPTR
A764 FF A7 D4           STX     TOPOS           TO POINTER
A767 CE B6 D0   SSRD05  LDX     #DOSFCB         DOS FCB ADDRESS
A76A B6 A7 FB           LDA A   SSUNIT          UNIT ADDRESS
A76D A7 02              STA A   XUN,X
A76F B6 A7 FD           LDA A   SSNPOS          NEXT TRACK POINTER
A772 B7 A7 EA           STA A   SSLPOS          LAST TRACK POINTER
A775 80 80              SUB A   #$80

A777 A7 1E              STA A   30,X
A779 B6 A7 FE           LDA A   SSNPO +1        NEXT SECTOR POINTER
A77C B7 A7 EB           STA A   SSLPOS+1        LAST SECTOR POINTER
A77F 80 40              SUB A   #$40
A781 A7 1F              STA A   31,X
A783 86 12              LDA A   #$12SR          READ SINGLE SECTOR
A785 A7 00              STA A   XFC,X
A787 BD B7 B6           JSR     DFM
A78A 27 04              BEQ     SSRD10
A78C BD B2 A9           JSR     ITYPDE          DISPLAY ERROR CODE
A78F 7E A7 F1           JMP     SSRD80          EXIT

A792 EE 26      SSRD10  LDX     38,X            GET NEXT ADDRESS
A794 FF A7 FD           STX     SSNPOS          SAVE NEXT TRACK/SECTOR POINTER
A797 CE B6 D0   SSRD15  LDX     #DOSFCB
A79A F6 A7 FC           LDA B   SSDPOS          NEXT DATA POSITION
A79D CB 2A              ADD B   #42             OFFSET TO DATA
A79F BD B2 A3           JSR     ZADDX           ADD B TO X
A7A2 FF A7 A6           STX     FRMPOS          STARTING FROM POSITION

A7A5 CE A7 A6   SSRD20  LDX     #FRMPOS         POINTER TO DATA IN SEEK RECORD
A7A8 8C B7 76           CPX     #DOSFCB+166 END OF DATA IN SECTOR
A7AB 26 0A              BNE     SSRD30
```

```
A7AD 7F A7 FC           CLR     SSDPOS          CLEAR DATA OFFSET
A7B0 7D A7 FD           TST     SSNPOS          IF ZERO, E N OF DATA
A7B3 27 3C              BEQ     SSRD80
A7B5 20 B0              BRA     SSRD05

A7B7 A6 00      SSRD30  LDA A   0,X
A7B9 81 00              CMP A   #$00            IS IT $00
A7BB 27 34              BEQ     SSRD80          YES, RESET & EXIT
A7BD 08                 INX                     BUMP
A7BE FF A7 A6           STX     FRMPOS          FROM POSITION
A7C1 7C A7 FC           INC     SSDPOS          BUMP DATA OFFSET
A7C4 81 1A              CMP A   #$1A            BYPASS DOS CHAR
A7C6 27 DD              BEQ     SSRD20
A7C8 81 02              CMP A   #$02            TEST FOR COMMAND LINE PREFIX
A7CA 26 07              BNE     SSRD35          IF NOT, BYPASS
A7CC 7C A7 F5           INC     DOSFLG          SET COMMAND LINE PRESENT
A7CF 86 2E              LDA A   #'.             LOAD COMMAND LINE ID
A7D1 20 09              BRA     SSRD38          BYPASS STORING THIS CHAR
A7D3 CE A7 D4   SSRD35  LDX     #TOPOS          DOS BUFFER POINTER
A7D6 A7 01              STA A   1,X             STORE LINE CHARACTER
A7D8 08                 INX                     BUMP
A7D9 FF A7 D4           STX     TOPOS           LINE BUFFER POINTER
A7DC 16         SSRD38  TAB                     SAVE A
A7DD BD B2 86           JSR     OUTEEE          SHOW THE CHARACTER
A7E0 C1 0D              CMP B   #$0D            IS IT END OF LINE?
A7E2 26 C1              BNE     SSRD20          IF NOT, GET NEXT CHAR
A7E4 86 0A              LDA A   #$0A            DISPLAY LINE AFTER C/R
A7E6 BD B2 86           JSR     OUTEEE
A7E9 CE A7 EA   SSRD50  LDX     #SSLPOS         RESET TRACK/SECTOR
A7EC FF A7 FD           STX     SSNPOS
A7EF 20 02              BRA     SSRD90

A7F1 7F A7 FF   SSRD80  CLR     PROCFG          RESET PROC COUNTER
A7F4 39         SSRD90  RTS
```

```
*
*  WORKING STORAGE / LINKAGE / EQUATES
*

A7F5 00         DOSFLG  FCB     0               DOS COMMAND LINE PRESENT FLAG

A7F6                    ORG     $A716
A7F6 00         SSUNI2  FCB     0               ALT UNIT NUMBER
A7F7 00         SSDP02  FCB     0               ALT DATA OFFSET
A7F8 00 00      SSNPO2  FDB     0               ALT TRACK/SECTOR
A7FA 0D         PROCF2  FCD     0               ALT PROC PRESENT F AG

A7FB 00         SSUNIT  FCB     0               UNIT NUMBER
A7FC 00         SSDPOS  FCB     0               DATA OFFSET
A7FD 00 00      SSNPOS  FDB     0               TRACK/SECTOR
A7FF 00         PROCFG  FCB     0               PROC PRESENT FLAG

                ****** DOS LINKAGE ******
B2E5                    ORG     $B2E5
B2E5 BD A7 2F           JSR     JCPROC          JOB CONTROL PROC LINKAGE

B2E8                    ORG     $B2EF           *
B2EF A7 00              FDB     UCTBL           USER COMMAND TABLE ADDRESS
```

```
A7EA            SSLPOS  EQU     SSRD50+1
A7A6            FRMPOS  EQU     SSRD20+1
A7D4            TOPOS   EQU     SSRD35+1
B000            LB      EQU     $B000
B2FC            LINPTR  EQU     $B2FC
B6D0            DOSFCB  EQU     $B6D0
0000            XFC     EQU     0
0002            XUN     EQU     2               UNIT NUMBER
0012            SSSR    EQU     18              SINGLE SECTOR READ
B6AE            INVAL   EQU     $B6AE
B283            ZWARMS  EQU     $B283
B2A9            ZTYPDE  EQU     $B2A9
B2A3            ZADDX   EQU     $B2A3
B7B6            DFM     EQU     $B7B6
B2A0            ZGETHN  EQU     $B2A0
B29D            ZDIE    EQU     $B29D
B2B5            ZLINEI  EQU     $B2B5
B286            OUTEEE  EQU     $B286

A042            STACK   EQU     $A042

                NAM     PROC    EXTRACT/CREATE PROC
                OPT     PAG,NOB
B080                    ORG     $B080
```

```
*  SYSTEM.....SWT /DOS68.42A
*  DATE WRITTEN.....AUG 8, 1979
*  AUTHOR.....GARY A. MAGNUSEN

*****************************************************
*       THIS PROGRAM LOADS INT THE T.C.A.I. WHEN CALLED
*****************************************************
```

```
B080            PROC    EQU     *
B080 FE B2 FC           LDX     LINPTR          DOS LINE BUFFER POINTER
B083 A6 00              LDA A   0,X             GET FIRST CHARACTER
B085 81 0D              CMP A   #$0D            IS FILE NAME THERE
B087 26 35              BNE     PROC50          FILE SPECS PRESENT
B089 8D B1 8C           JSR     WROPEN          OPEN FILE TO CREATE A PROC FILE
B08C 7E B0 D7           JMP     ESCAPE

B08F 86 2E      PROC10  LDA A   #$2E             LOAD COMMAND ID
B091 7D B1 71           TST     DOSFLG          IF COMMAND PRESENT?
B094 26 02              BNE     PROC20          BYPASS
B096 86 20              LDA A   #$20            LOAD DATA ID
B098 BD B2 A6   PROC20  JSR     OUTEEE          SHOW INPUT TYPE
B09B BD B2 B5           JSR     ZLINEI          READ A LINE
B09E FE B2 FC           LDX     LINPTR          DOS LINE BUFFER POINTER
B0A1 A6 01              LDA A   1,X             ARE WE CHANGING COMMAND/DATA TYPES
B0A3 81 1B              CMP A   #$1B            IF (ESC),
B0A5 27 30              BEQ     ESCAPE          GO REVERSE TYPES
B0A7 81 0D              CMP A   #$0D            IS THIS THE LAST LINE?
B0A9 27 08              BEQ     PROC40
B0AB B6 B1 71           LDA A   DOSFLG          LOAD LINE TYPE
```

```
B0AE BD B1 3D          JSR    WRIT40    GO WRITE LINE
B0B1 20 1C             BRA    PROC10    GET NEXT LINE

B0B3 BD B1 52  PROC60  JSR    WRCLOSE   CLOSE PROC FILE
B0B6 BD B1 5F          JSR    CLRFCB    INITIZ FCB
B0B9 BD B0 19          JSR    RDOPEN    GO OPEN PROC FILE
B0BC 20 03             BRA    PROC70    ENTER COMMAND LOOP

B0BE BD B0 F1  PROC50  JSR    RDOPEN    OPEN PROC FILE
B0C1 CE B6 D0  PROC70  LDX    #DOSFCB   DOS FCB
B0C4 A6 02             LDA A  XUN,X     SAVE
B0C6 B7 A7 FB          STA A  SBUNIT    UNIT NUMBER
B0C9 EE 0E             LDX    14,X      SAVE
B0CB FF A7 ID          STX    SSNPOS    TRACK/SECTOR
B0CE BD B1 06          JSR    RDCLOSE   GO CLOSE PROC FILE
B0D1 7C A7 FF          INC    PROCFC    SET PROC PRESENT FLAG
B0D4 7E B2 83          JMP    ZWARMS    GO TO DOS

B0D7                   ESCAPE EQU    *
B0D7 7D B1 71          TST    DOSFLG    TEST COMMAND TYPE FLAG
B0DA 27 08             BEQ    EXC30     IF ZERO, SET TO COMMAND LINE
B0DC 7F B1 71          CLR    DOSFLG    SET TO DATA LINE
B0DF CE B1 B1          LDX    #DTAMES   LOAD DATA LINE MESSAGE
B0E2 20 08             BRA    EXC50     GO DISPLAY

B0E4 86 02     EXC30   LDA A  #$02      SET TO COMMAND LINE
B0E6 B7 B1 71          STA A  DOSFLG
B0E9 CE B1 9D          LDX    #DOSMES   LOAD COMMAND LINE MESSAGE
B0EC BD B2 A6  EXC50   JSR    ZOUTST    GO DISPLAY MESSAGE
B0EF 20 9E             BRA    PROC10    BRANCH TO READ LINES


B0F1                   RDOPEN EQU    *         OPEN PROC FILE
B0F1 CE B6 D0          LDX    #DOSFCB
B0F4 BD B2 91          JSR    ZFLSPC    GET FILE SPECS
B0F7 25 1A             BCS    EREXIT    ERROR/RETURN TO DOS
B0F9 CE B6 D0  RDOP10  LDX    #DOSFCB   READ FCB POINTER
B0FC 86 04             LDA A  #$04R     SET READ SEQ FILE FC
B0FE A7 00             STA A  XFC,X
B100 BD B7 86          JSR    DFM       DISK FILE MANAGEMENT
B103 26 0E             BNE    EREXIT    ERROR/RETURN TO DOS
B105 39                RTS


B106 CE B6 D0  RDCLOSE LDX    #DOSFCB
B109 86 06             LDA A  #QSWC     CLOSE FC
B10B A7 00             STA A  XFC,X
B10D BD B7 8          JSR    DFM       DISK FILE MANAGEMENT
B110 26 01             BNE    EREXIT    ERROR/RETURN TO DOS
B112 39                RTS

B113 BD B2 A9  EREXIT  JSR    ZTYPDE    DISPLAY ERROR CODE
B116 BD B7 83          JSR    C FM      CLOSE ALL FILES
B119 7E B2 83          JMP    ZWARMS    RETURN TO DOS

               **** CREATE PROC ****

B11C CE B1 72  WROPEN LDX    #WROMES   OPEN FILE NAME
B11F BD B2 A6          JSR    ZOUTST
B122 BD B2 B5          JSR    ZLINEI    ENTER FILE NAME IN DOS BUFFER
B125 CE B6 D0          LDX    #DOSFCB
B128 BD B2 91          JSR    ZFLSPC    GET FILE SPECS
B12B 25 E6             BCS    EREXIT    ERROR
B12D CE B6 D0          LDX    #DOSFCB   DOS FCB
B130 86 01             LDA A  #QSO4W    SET WRITE SEQ FILE FC
B132 A7 00             STA A  XFC,X
B134 BD B7 86          JSR    DFM       DISK FILE MANAGEMENT
B137 26 DA             BNE    EREXIT    ERROR
B139 39                RTS
B13A BD B2 97  WRITE   JSR    ZGNCHR    GET I/O CHAR
B13D 81 00     WRIT40  CMP A  #$00       IF NULL$?
B13F 27 F9             BEQ    WRITE     BYPASS
B141 CE B6 D0          LDX    #DOSFCB   DOS FCB
B144 C6 02             LDA B  #QSWRIT   SET WRITE FC
B146 E7 00             STA B  XFC,X
B148 BD B7 86          JSR    DFM
B14B 26 C6             BNE    EREXIT    ERROR/RETURN TO DO
B14D 81 0D             CMP A  #$0D      IS THIS EOR
B14F 26 E9             BNE    WRITE
B151 39        WRIT90  RTS               EXIT

B152 CE B6 D0  WRCLOSE LDX    #DOSFCB
B155 86 03             LDA A  #QSWC     CLOSE FC
B157 A7 00             STA A  XFC,X
B159 BD B7 8A          JSR    DFM       DISK FILE MANAGEMENT
B15C 26 B5             BNE    EREXIT    ERROR/RETURN TO DOS
B15E 39                RTS               EXIT


B15F CE B6 D0  CLRFCB LDX    #DOSFCB   INITIZ
B162 C6 0E             LDA B  #14       FCB
B164 4F                CLR A             FOR READ
B165 A7 00             STA A  0,X        AFTER WRITE
B167 A7 01             STA A  1,X
B169 6F 0C     CLRF10  CLR    12,X
B16B 08                INX
B16C 5A                DEC B
B16D 26 FA             BNE    CLRF10
B16F 39                RTS
               *
               * WORKING STORAGE & EQUATES
               *
B170 00        LSTCHR  FCB    0
B171 00        DOSFLG  FCB    0
B172 0D        WROMES  FCB    $0D,$0A
B173 0A
B174 45                FCC    'ENTER (UNIT:NAME.EXT) OF PROC TO CREATE
B175 4E 54     EXAMPLE OF CREATING AND USING A PROCEDURE
B177 45 52
B179 20 28     &PROC
B17B 55 4E
B17D 49 54     ENTER (UNIT:NAME.EXT) OF PROC TO CREATE 1:LETTER.PRC
B17F 3A 4E     ENTERING COMMANDS
B181 41 4D     .DELETE 1:LETTER.OBJ
B183 45 2E     .EDIT 1:LETTER.SOU
B185 45 58     .ASSEMB 1:LETTER.SOU,1:LETTER.OBJ,,N,N
B187 54 29
B189 20 4F
B18B 46 20
B18D 50 52
B18F 4F 43
B191 20 54
B193 4F 20
B195 43 52
B197 45 41
B199 54 45
B19B 20
B19C 00                FCB    $00
B19D 45        DOSMES  FCC    'ENTERING COMMANDS'
B19E 4E 54
B1A0 45 52
B1A2 49 4E
B1A4 47 20
B1A6 43 4F
B1A8 4D 4D
B1AA 41 4E
B1AC 44 53
B1AE 0D                FCB    $0D,$0A,$00
B1AF 0A 00
B1B1 45        DTAMES  FCC    'ENTERING DATA'
B1B2 4E 54
B1B4 45 52
B1B6 49 4E
B1B8 47 20
B1BA 44 41
B1BC 54 41
B1BE 0D                FCB    $0D,$0A,$00
B1BF 0A 00

               *
               *       (PROCA) COMMON WORKING STORAGE
               *

A7F6                   ORG    $A7F6
A7F6 00        SBUNI2  FCB    0         ALT UNIT NUMBER
A7F7 00        SSDPO2  FCB    0         ALT DATA OFFSET
A7F8 00 00     SSNPO2  FDB    0         ALT TRACK/SECTOR
A7FA D0        PROCF2  FCB    0         ALT PROC PRESENT FLAG

A7FB 00        SBUNIT  FCB    0         UNIT NUMBER
A7FC 00        SSDPOS  FCB    0         DATA OFFSET
A7FD 00 00     SSNPOS  FDB    0         TRACK/SECTOR
A7FF 00        PROCFC  FCB    0         PROC PRESENT FLAG

               *       DOS EQUATES

B6D0           DOSFCB  EQU    $B6D0
B2FC           LINPTR  EQU    $B2FC
B283           ZWARMS  EQU    $B283
B286           OUTEEE  EQU    $B286
B291           ZFLSPC  EQU    $B291
B297           ZGNCHR  EQU    $B297
B2A6           ZOUTST  EQU    $B2A6
B2A9           ZTYPDE  EQU    $B2A9
B2B5           ZLINEI  EQU    $B2B5

B7             CDFM    EQU    $B7A3
B786           DFM     EQU    $B786

0000           XFC     EQU    0         FUNCTION CODE
0001           XERR    EQU    1         ERROR STATUS
0002           XUN     EQU    2         UNIT NUMBER

0001           QSO4W   EQU    1         OPEN FOR WRITE (CREATE)
0002           QSWRIT  EQU    2         WRITE DATA
0003           QSWC    EQU    3         CLOSE FOR WRITE
0004           QSO4R   EQU    4         OPEN FOR READ
0006           QSRC    EQU    6         CLOSE FOR READ
                       END
      NO ERROR(S) DETECTED
```

EXAMPLE OF USING A CREATED PROCEDURE

```
&PROC 1:LETTER.PRC

&.DELETE 1:LETTER.OBJ

&.EDIT 1:LETTER.SOU
*1C.LETTER.LETTER.
1.00*          NAM     LETTER
*E

&.ASSEMB 1:LETTER.SOU,1:LETTER.OBJ,,N,N
        NO ERROR(S) DETECTED

&.GET 1:LETTER.OBJ

&.PAUSE   ENTER PATCHES HERE
&

&.GO 100

&.DELETE 1:LETTER.IN

&.RENAME 1:LETTER.OUT,1:LETTER.IN

&


.GET 1:LETTER.OBJ
.PAUSE   ENTER PATCHES HERE
.GO 100
.
ENTERING DATA
 1:LETTER.IN
 1:LETTER.OUT

.ENTERING COMMANDS
.DELETE 1:LETTER.IN
.RENAME 1:LETTER.OUT,1:LETTER.IN
.
```

```
&.DELETE 1:LETTER.OBJ

&.EDIT 1: ETTER.SOU
*1C.LETTER,LETTER.
1.00*          NAM     LETTER
*E

&.ASSEMB 1:LETTER.SOU,1:LETTER.OBJ,,N,N
        NO ERROR(S) DETECTED

&.GET 1:LETTER.OBJ

&.PAUSE   ENTER PATCHES HERE
&

&.GO 100

&.DELETE 1:LETTER.IN

&.RENAME 1:LETTER.OUT,1:LETTER.IN
```

EXPANDED CMDS. CCS BASIC

Jeffery Brownstein
2 Tor Rd.
Wappingers, NY 12590

Note: Locations in parenthesis are for version 4.3 but are not tested.

1-decide how much room is needed.
2-move the number build buffer and for-next buffer (from above table).
3-move step and to commands to new top of table- they malfunction if left in the middle. Reason unknown.
4-place a large dummy command between last old command and to command.
5-change certain references to the command table.
6-move beginning of basic source code (loc 109Hex) upward.

Various revisions of CSS Cassette Basic will have locations slightly offset from those presented. My changes to Version 4.0 have been running without problems for over a year.

When my number build buffer and for-next buffer were moved away there was empty space up to 21B3 (20A6). 21B4 (20A7) is the first location used by the cassette data files. For this reason I ended the table at 21B3.

The new locations for the above mentioned buffer areas became 24DB to 25DB. Note the changes:

At 0B03 (0B05) change CE 20B6 (20A9) to CE 24DB Numb. Build
At 0AD2 (0AD4) change 2134(2127) to 255B For-next
At 1405 (13F8) change 2134(2127) to 255B "
At 14C3 (14BB) change 2134(2127) to 255B "
At 14C8 (none) change 2134(none) to 255B "
At 142C (none) change 21B4(none) to 25DB (tests for max. 8 nested)

Be certain to make the For-next value 2124 (2127) the same number of bytes above the buffer start as before. For me, 255B was up the correct distance from 24DB. The old buffer started 20B6(20A9).

Old TO and STEP		New TO and STEP	
20A9	(209C)	21A7	
20AA 54	T	21A8 54	T
20AB 4F	O	21A9 4F	O
20AC 00		21AA 00	
20AD 0F	(0EFB)	21AB 0F	
20AE 08		21AC 08	
20AF 53	S	21AD 53	S
20B0 54	T	21AE 54	T
20B1 45	E	21AF 45	E
20B2 50	P	21B0 50	P
20B3 00		21B1 00	
20B4 0F	(0EFB)	21B2 0F	
20B5 08		21B3 08	End of table.
20B6	Begins For-next and number build buffers.	21B4	Begins file buffer.

At 09CB(09CB)change CE 20A9 (209C) to CE 21A7 Loc. before TO
At 0A02(0A02)change 8C 20B3 (20A6) to 8C 21B1 Third from end
AT 1455(1448)change 8C 20AD (20A0) to 8C 21AB Ninth from end
At 1470(none)change 8C 20B4 (none) to 8C 21B2 Next to last

The dummy command to fill up the big new command space is eroded as you add new commands. It looks like: 40 40 40-----40 40 00 FF FF

LINPUT FOR CSS BASIC VERSIONS 4.0 and 4.3

After discovering that location 0E55 (0E48 in ver. 4.3) controls whether the INPUT statement will allow commas to be imbedded in the inputted text I was successful in disabling the test whenever desired. Two approaches were tried; A) using POKE
B) new command

A)
```
10 POKE (3669,0)      (Use 3656 for version 4.3)
20 INPUT A$
30 POKE (3669,9)
```

B)
```
7F 0E55   Clear byte
BD 0E7E   Execute input
C6 09     Restore byte
F7 0E55
39        Return
```

An entry in the command table will be needed. (Ver. 4.3 use 0E48 . Input is at 0E71)

RESTORE DATA POINTER TO LINE NUMBER DESIRED
CSS BASIC VERSION 4.0

First change the command table entry for RESTORE (at 1FD2) to point to the new routine. Leave the old routine alone as we will use it if no line number has been specified.

```
BD 0961   read byte after word RESTORE
26 03     end of line?
7E 1079   goto old restore command
81 3A     multiple statement/line?
27 F9     branch
BD 096E   get line #
BD 0933   find line
24 05     test for no such line
C6 07     load error 7
7E 0B5A   process error msg.
09
DF 44
39        return
```

PERCOM
PERCOM DATA COMPANY, INC.

PERCOM PLUG-IN ADAPTER FOR TRS-80* AND SWTP MP-F MINI-FLOPPY DISK CONTROLLERS MAKES DATA SEPARATION FUNCTION RELIABLE, REDUCES DATA READ ERRORS

Garland, Texas - August 21, 1979 - Harold Mauch, president of Percom Data Company, announced here today that the company is now offering a plug-in adapter for the TRS-80* and Southwest Technical Products' MP-F mini-disk controllers which virtually eliminates the data read errors caused when clock and data bits are not reliably separated during playback.

The problem relates to the higher storage density of the inner disk tracks, and is not uncommon with either controller.

Called the SEPARATOR†, the adapter may be installed without making any changes to the host system. The user merely removes the 1771 disk controller IC from the host controller, installs the IC in the DIP socket on the SEPARATOR† card and then plugs the card into the vacated 1771 socket of the host system.**

The SEPARATOR† uses an external data separation circuit, one, which Mauch explained, that features special circuitry to compensate for a phenomenon referred to as "bit shifting."

Bit shifting arises during playback (reading) of high density data. Mauch said this is the source of most read errors.

An assembled and tested SEPARATOR† adapter sells for $29.95, including installation instructions.

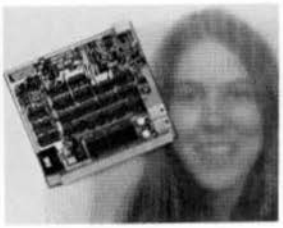
Orders may be placed by calling Percom toll-free at 1-800-527-159 and may be paid by check or money order, COD or charged to a Visa or Master Charge account. Texas residents must add 5% sales tax.

Dealer inquiries are invited.

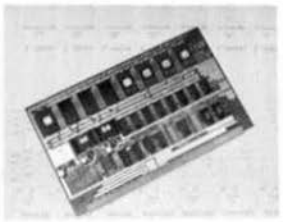
† Trademark of Percom Data Company, Incorporated.

* Trademark of Tandy Radio Shack which has no relationship to Percom Data Company.

** Opening the Expansion Interface of the TRS-80*, which is necessary to access the disk controller, may void the limited 90-day warranty.



DS-68 DIGISECTOR



PSB-08 PROM SYSTEM BOARD

B-08 EPROM PROGRAMMER

UIO UNIVERSAL I/O BOARD

INNOVATIVE PRODUCTS FOR 6800 USERS

DS-68 DIGISECTOR is a random access video digitizer featuring 256 × 256 picture element scan and 64 levels of grey scale, with conversion times as low as 3 microseconds per pixel. It accepts either interlaced (NTSC) or non-interlaced (industrial) video input. Use it for computer portraiture, moving target indicators, precision security systems, fast to slow scan conversion. . . with clever software, the Digisector can read just about anything. Truly a professional tool at a price you can afford. $169.95

DM-85 DISK MIXER is an add-on board for the Smoke Signal Broadcasting BFD-68A Disk Controller which allows operation of both 8" and 5" drives. Controller mode (8" or 5") is selected on a drive-by-drive basis, so any mix of 5" and 8" drives is allowable. The 2" × 3" PC board mounts inconspicously on the back of the BFD-68A. Its operation is completely transparent to software. An oscilloscope is required for the setup procedure. Kit Price: $39.95

B-08 2708 EPROM PROGRAMMER is a compact unit that fits in the 6800's I/O slot. A safety switch and LED indicator provide control over the high programming voltage generated on board. An industrial quality Textool socket and extended board height allow effortless PROM insertion and retrieval. Fully commented source listings of U2708 is included in the Owner's Manual. $99.95

U2708 utility for testing, burning, verifying and copying 2708s in EPROM. $29.95

M6809 EMULATOR is a machine language program that will emulate all of the functions of the Motorola 6809 third generation microprocessor. Developed for use on any 6800 system, the program allows software development and debugging. The 3K byte program is complete with a 6809 mini-monitor and single-step trace routines. Fully commented source listing included. Specify Smoke Signal Broadcasting or FLEX™ disk, or KCS cassette. $49.95

PSB-08 PROM SYSTEM BOARD features 1K of high speed, low-power RAM and space for up to 8 2708 EPROMs, both DIP-switch addressable to start on any 8K boundary in memory. The exclusive I/O select feature allows you to move I/O locations up to any unused 1K block in the EPROM memory space. This permits memory expansion to a full 56K of contiguous user RAM. $119.95

UIO UNIVERSAL I/O BOARD helps you with your custom interfaces. It has space for a 40-pin wire wrap socket into which you may plug any of Motorola's 40 or 24-pin interface chips. All data and control lines are connected to the appropriate edge connector pins. All other bus connections are brought out to a 16-pin socket pad. +5 volt regulator and all Molex connectors are provided; regulated +5 and ground are bused among the locations for up to 35 14-pin ICs. $24.95

THE MICRO WORKS P.O. BOX 1110, DEL MAR, CA 92014 714-756-2687

Inventory Problems?

Are you having trouble keeping the right nuts ond bolts in stock? Since even o simple mistake con cost you time and money, a good inventory system should do more than just count parts. It should tell you exactly what you need, when you need it, where to get it, and how much it will cost.

The MSI Inventory System Seven enables you ro maintain o versatile dora base for controlling Inventory. It lists port number, description, quantity on hand, vendor, cost, selling price, optional pricing, usage levels for previous month, present month, ond year-to-dare, and much more.

When quantity on hand items reach minimum levels, the System Seven compiles on automatic reorder list. This list can be generated by specific vendor as well as a complete listing of all materials to be ordered.

In addition to the item listing, the inventory System Seven "bill of materials" provides you with a complete inventory of items used in the manufacture of subassemblies and complete products. It also contains other cost items such as labor costs, total raw materials costs, and miscellaneous costs.

The MSI Inventory System Seven is built around the versatile MSI 6800A Computer with 56K of RAM. An integral dual mini-floppy memory gives you an additional 630K of memory and makes inventory control fast and efficient. The System Seven will interface with any industry standard CRT, and you have the option of both a "daisy wheel" word processor for high quality document preparation and o dot matrix printer for high speed production.

The System Seven can be expanded to handle all your data processing needs or you can select one of nine other MSI systems now available for business, industrial, scientific, educational, and personal applications.

If you need more than just a nuts and bolts inventory system, we have more information about how the Inventory System Seven can solve your problems economically.

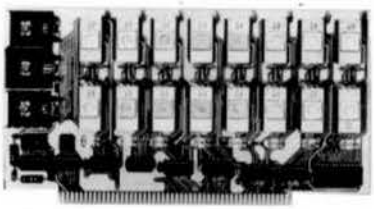
MSI Inventory System Seven

MSI

Midwest Scientific

220 W. Cedar, Olathe, Kansas 66061, (913) 764-3273
TWX 910 749 6403 (MSI OLAT), TELEX 42525 (MSI A OLAT)

16K EPROM CARD-S 100 BUSS



$59.95 KIT

OUR BEST SELLING KIT!

USES 2708's!

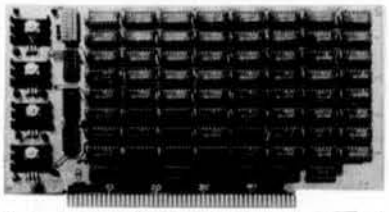
Thousands of personal and business systems around the world use this board with complete satisfaction. Puts 16K of software on line at **ALL TIMES!** Kit features a top quality soldermasked and silk-screened PC board and first run parts and sockets. All parts (except 2708's) are included. Any number of EPROM locations may be disabled to avoid any memory conflicts. Fully buffered and has WAIT STATE capabilities.

OUR 450NS 2708'S ARE $8.95 EA. WITH PURCHASE OF KIT

ASSEMBLED AND FULLY TESTED ADD $25

8K LOW POWER RAM KIT-S 100 BUSS

SALE



PRICE CUT!

$119.50 KIT

(450 NS RAMS!)

Thousands of computer systems rely on this rugged, work horse, RAM board. Designed for error-free, NO HASSLE, systems use.

KIT FEATURES:

1. Doubled sided PC Board with solder mask and silk screen layout. Gold plated contact fingers.
2. All sockets included.
3. Fully buffered on all address and data lines.
4. Phantom is jumper selectable to pin 67.
5. FOUR 7805 regulators are provided on card.

Blank PC Board w/Documentation **$29.95**
Low Profile Socket Set...**13.50**
Support IC's (TTL & Regulators) **$9.75**
Bypass CAP's (Disc & Tantalums) **$4.50**

ASSEMBLED AND FULLY BURNED IN ADD $30

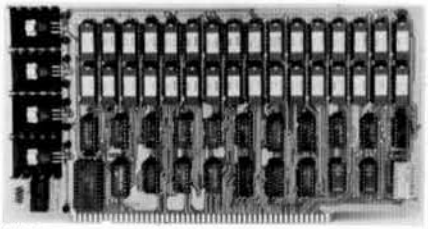
16K STATIC RAM KIT-S 100 BUSS

PRICE CUT!

$279 KIT

FOR 250NS ADD $10

FULLY STATIC, AT DYNAMIC PRICES



WHY THE 2114 RAM CHIP?

We feel the 2114 will be the next industry standard RAM chip (like the 2102 was). This means price, availability, and quality will all be good! Next, the 2114 is FULLY STATIC! We feel this is **the ONLY** way to go on the S-100 Buss! We've all heard the HORROR stories about some Dynamic Ram Boards having trouble with DMA and FLOPPY DISC DRIVES. Who needs these kinds of problems? And finally, even among other 4K StaticRAM's the 2114 stands out! Not all 4K static Rams are created equal! Some of the other 4K's have clocked chip enable lines and various timing windows just as critical as Dynamic RAM's! Some of our competitor's 16K boards use these "tricky" devices. But not us! The 2114 is the ONLY logical choice for a trouble-free, straightforward design.

KIT FEATURES:

1. Addressable as four separate 4K Blocks.
2. ON BOARD BANK SELECT circuitry. (Cromemco Standard). Allows up to 512K on line!
3. Uses 2114 (450NS) 4K Static Rams.
4. ON BOARD SELECT ABLE WAIT STATES.
5. Double sided PC Board, with solder mask and silk screened layout. Gold plated contact fingers.
6. All address and data lines fully buffered.
7. Kit includes ALL parts and sockets.
8. PHANTOM is jumpered to PIN 67.
9. LOW POWER: under 2 amps TYPICAL from the +8 Volt Buss.
10. Blank PC Board can be populated as any multiple of 4K.

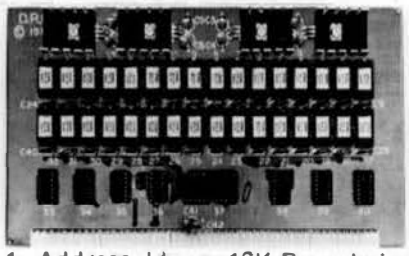
BLANK PC BOARD W/DATA—$33
LOW PROFILE SOCKET SET—$12 ASSEMBLED & TESTED—**ADD $30**
SUPPORT IC'S & CAPS—**$19.95**

16K STATIC RAM SS-50 BUSS

PRICE CUT!

$275 KIT

FULLY STATIC AT DYNAMIC PRICES



KIT FEATURES:

1. Addressable on 16K Boundaries
2. Uses 2114 Static Ram
3. Runs at Full Speed
4. Double sided PC Board. Solder mask and silk screened layout. Gold fingers.
5. All Parts and Sockets included
6. Low Power: Under 2 Amps Typical

FOR SWTPC 6800 BUSS!

ASSEMBLED AND TESTED - $30

BLANK PC BOARD—$33 COMPLETE SOCKET SET—$12
SUPPORT IC'S AND CAPS—**$19.95**

S-100 Z80 CPU CARD

ASSEMBLED AND TESTED! READY TO USE! Over 3 years of design efforts were required to produce a TRUES-100 Z80 CPU at a genuinely bargain price! **4 MHZ! $159.95**

FEATURES:

- 2 or 4 MHZ Operation.
- Generates MWRITE, so no front panel required.
- Jump on reset capability
- 8080 Signals emulated for S-100 compatability.
- Top Quality PCB, Silk Screened, Solder Masked, Gold Plated Contact Fingers.

Perfect For OEM's

LOW POWER - 250NS
2114 RAM SALE!

4K STATIC RAM'S. MAJOR BRAND. NEW PARTS.

These are the most sought after 2114's, LOW POWER and 250NS FAST.

SPECIAL SALE: $7.50 ea. or 8 For $55

(We reserve the right to limit quantities.)

PROC. TECH. QUITS THE MICROPROCESSOR BUSINESS!
FACTORY CLOSE OUT - SPECIAL PURCHASE!
#16KRA

16K S-100 Dynamic Ram Board - $149.95

ORIGINALLY PRICED AT $429 each!

We purchased the remaining inventory of PT's popular 16K Ram Board when they recently closed their plant. Don't miss the boat! These are brand new, fully tested, ASSEMBLED and ready to go. All are sold with our standard 90 day limited warranty!!

72 Page Full Manual, Included Free!

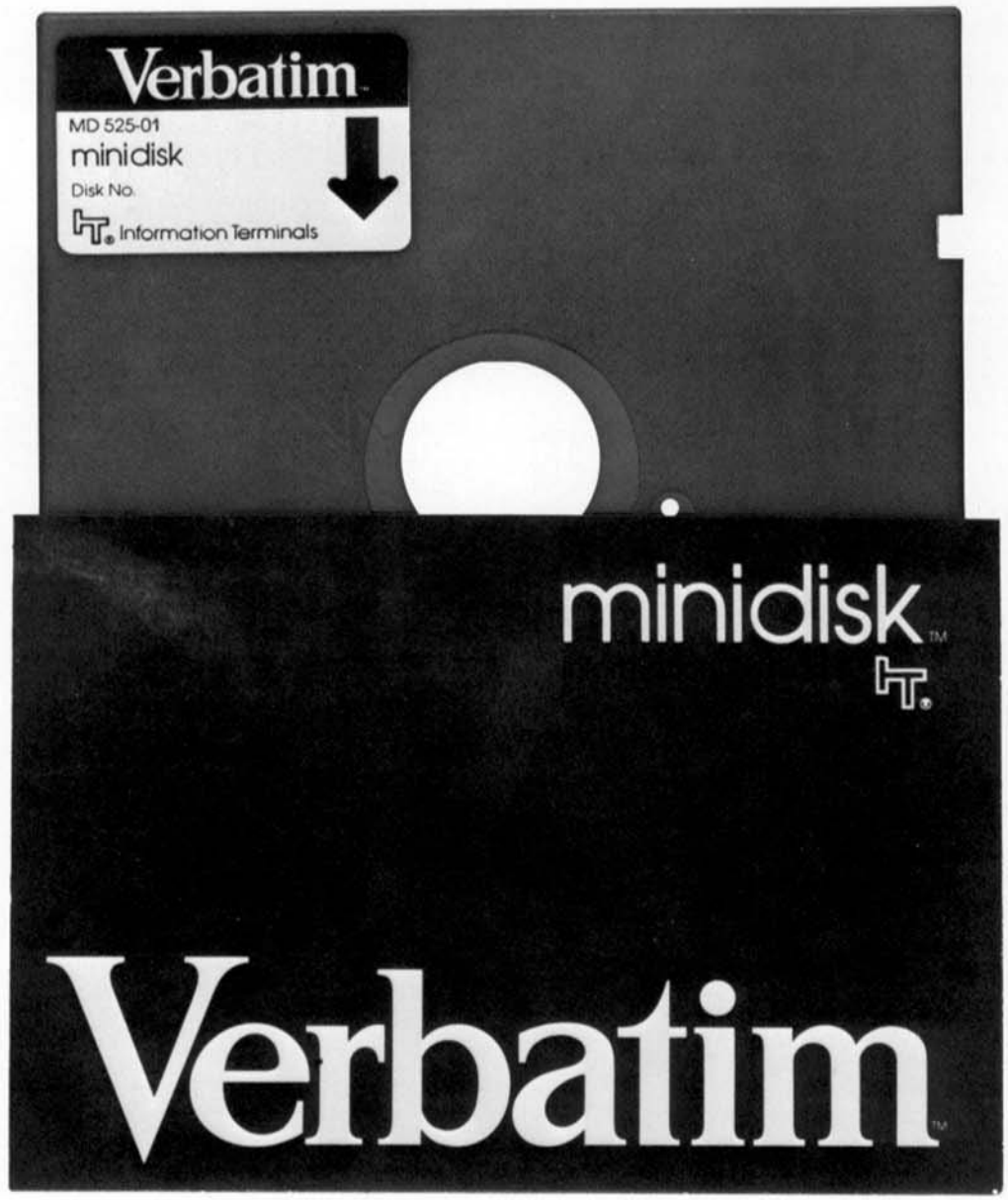
NOT ASSOCIATED WITH DIGITAL RESEARCH OF CALIFORNIA, THE SUPPLIERS OF CPM SOFTWARE.

Digital Research: Computers
(OF TEXAS)

P.O. Box 401565 • GARLAND, TEXAS 75040 • (214) 494-1505

TERMS: Add 50¢ postage, we pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa, MasterCharge, and American Express cards. Tex. Res. add 5% Tax. Foreign orders (except Canada add 20% P & H. 90 Day Money Back Guarantee on all items.

5-1/4" Minidisk — Soft or Hard Sector



SAVE DISK

SOUTH EAST MEDIA SUPPLY

6131 Airways Blvd. 615-892-1328
Chattanooga, TN 37421

Software Source Books™

Combining detailed descriptions with complete source listings, these books explain the internal operations and algorithms used in Hemenway Associate's popular systems software.

How much would such a complete software resource cost? If you've seen the PAPERBYTE books by Jack Hemenway and Robert Grappel you know how inexpensive they can be. And now you can have the companion volumes to the RA6800ML macro assembler and LINK68 linking loader books.

Remember, these are not just books; they are SoftwareSourceBooks ---- complete Software resources! Order them today; VISA and MasterCharge accepted.

CP/68™ OPERATING SYSTEM
($34.95)

* PIP Peripheral Interchange
* Program transfers data between physical devices
* Wildcard Filenames and Extensions
* Relocatable anywhere in Memory
* Extended Instruction set includes 6809-type instructions (PSHX, PULX, etc)
* Device-independent I/O
* Random and Sequential Files
* Fits in less than 8K
* Chaining and overlaying
* Single Supervisor Call furnishes all DOS services
* Easily interfaced to new devices and peripherals
* Dynamic file allocation

STRUctured BAsic Language (STRUBAL+™) COMPILER for both business and scientific uses
($49.95)

* Variable precision from 4 to 14 digits
* Structured Programming forms
* Produces Relocatable and linkable code
* COMMON and DUMMY sections
* Extensibility
* String Handling
* Full scientific package
* Data structures with mixed data types

XA6809 Macro Linking Cross Assembler
($24.95)

* Runs on any M6800
* Full Macro facilities
* COMMON section for the production of ROMable code Conditional Assembly
* Generates linkable and relocatable code
* Sorted Symbol table listing
* Hash-coded Symbol table for speed

==========

Hemenway Associates Inc. 101 Tremont St. Boston MA 02108

Name _____ Title _____ Company _____

Street _____ City _____ State _____ Zip _____

( ) Check enclosed in the amount of $.....

( ) Bill VISA ( ) Bill MasterCharge

Card No..... Exp. Date.....

Please send the following books:

_____ _____ _____

Add $0.75 per book to cover postage and handling

NOW INDUSTRIAL QUALITY AT LOW COST

FROM THOMAS INSTRUMENTATION

Industrial system boards are now available separately for OEM, prototyping or hobbyist applications. Shipped from stock, these are the same quality cards used in monitors and machine tool controls designed for GM, LTI, and General Electric. All cards are SS-50 buss compatible and are suitable for dedicated applications. The CPU card and the Video RAM Card may be combined on a TI backplane as a stand-alone micro — ideal for prototypes or hobbyists. ATTENTION OEM's: If you have a control data acquisition, monitoring, or other microprocessor application. Check with TI for more information about custom software design for the TI CPU or any other 6800 series system. TI also has non-SS-50 buss single board 6800 systems.

SS-50 SUPER CPU

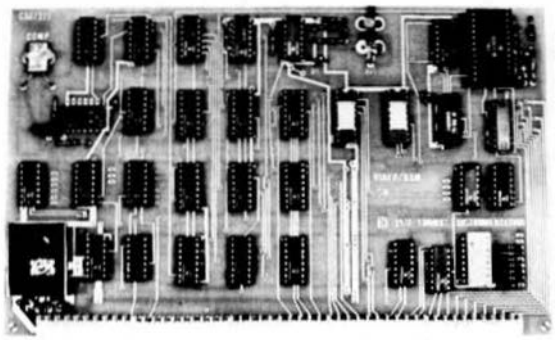
- SS-50 or stand alone computer
- 1K of RAM at $A000 I/O on board at $A400 (Relocatable)
- 2K Monitor (Mikbug compatible) in 2708 EPROM
- 2 8 bit parallel ports with 2 control bits and power
- RS-232 ACIA port. 2nd TTL ACIA optional
- 3 16 bit counter/timers (expandable to 6 add 2nd 6840)
- 128 byte RAM at 0000 is jumper selectable
- Battery back-up for 32 bytes of RAM
- Plug back to back with Video RAM for Stand alone micro or customized smart terminal

ASSEMBLED $195.00
CARD AND DOCUMENTATION $49.00

SS-50 VIDEO RAM

- Fully synchronous operation — No jitter
- 7 by 9 Characters Programmable reverse video
- Full 128 Character ASCII set
- 1K of memory can be mapped to any 1K boundary
- Full documentation includes software (Replaces OUTEE)

ASSEMBLED $149.00
CARD, CRYSTAL and DOCUMENTATION $39.00



TI SS-50 Wire-Wrap Card	**24.00**
TI SS-50 Parallel I/O Card	**95.00**
Card only	**35.00**
GIMIX 16K Static RAM w/Soft addressing	**359.00**
GIMIX 16 RAM without Soft addressing	**290.00**
3, 4, 7 SLOT Backplanes (per slot)	**4.00**

TI cards available from stock

THOMAS INSTRUMENTATION

188-8th Street, Avalon, N.J. 08202
Phone (609) 967-4280

VISA

DEALERS FOR GIMIX SWTPC SSB

CALL FOR DEALER, OEM. AND QUANTITY PRICES

MINI-SYSTEMS

CASH-FLOW Bookkeeping System

This mini accounts payable and general ledger system provides account codes for categorization, checking account reconciliation, year-to-date totals, a dated purge facility, and complete activity recording. Reports include expense report, account report, all activity, written checks, outstanding checks, adjustments, and deposits. Sort keys have range selections. It is written in Random BASIC for SSB DOS.

introductory price **$99.95**

Does Your BASIC Have:

Random Records, PRINT USING, ON ERRor EDIT

. . . Our's DOES!

Random Access BASIC, written for the Smoke Signal Broadcasting disk system, allows the user to retrieve, change, or add a record directly within a file. Many proven application packages are available!

$99.95

MINI-SYSTEMS

RANDOM V2 INVENTORY SYSTEM

Keeps record of: Part Name/Number, Vendor Code, Selling Price, Average Cost, Quantity-on-Hand, Quantity-on-Order, Minimum Quantity, Category Code, Location Code. There are no system limits to the number of items. Reports may be sorted by Part Name/Number, Category, Location, or Vendor and a reorder report is provided. On-line search and invoicing is included. It is written in Random BASIC for SSB DOS.

introductory price **$69.95**

COMPUTERWARE

6800 system experts

1512 Encinitas Blvd., Box 668
Encinitas, CA 92024

(714) 436-3512

Random V2 Mailing List System

Label printing directed to any port — optional phone number & special code—selectable label spacing —date field in each record—sorted reports or labels can be requested by range of values in fields: date, code, zip, name—title/country field in each record—all fields except date are alphanumeric — written in Random BASIC for SSB DOS.

introductory price
$89.95

LOOK TO US FOR HARDWARE TOO!!!

Smoke Signal Broadcasting, SWTPC, Anadex, Centronics, NEC, SOROC, Micro-Term, Sanyo, Ledex, Micro-Works. . . .

ARE YOU ON OUR MAILING LIST?

Fill out and send in the form.
If you are not in our mailing list, you will receive:

(1) A coupon worth $5 on your first Computerware purchase.

(2) All of our current product information.

$5 **$5**

Name

Street

City State Zip

System Description:

ONE COUPON PER CUSTOMER

$5 **$5**

SMOKE SIGNAL BROADCASTING PRESENTS...

THE SMOKE WRITER (VDB-1)

The SSB SMOKE WRITER incorporates the latest advances in electronic technology to bring you and the SS-50 bus a truly unique video display board. The SMOKE WRITER uses the MC6845 CRT Controller chip and provides total control over the display format.

IT'S NEW

$349.00

The standard features of the SMOKE WRITER are:

- *80 x 24 display with 32 graphic characters. Optional character generator ROM with 128 ASCII characters plus 128 graphic characters.*
- *Upper and lower case characters with lower case descenders.*
- *Programmable character set, a total of 128 characters in a 2K EPROM; a 256 character 4K EPROM is optional.*
- *1K EPROM for Software drivers.*
- *Reduced intensity or reversed video*
- *Programmable display rate (10 to 5000 character per second) equivalent to 100 to 50K baud.*
- *Protected fields.*
- *Addressable Cursor.*
- *2K video display RAM accessible by the CPU as standard RAM Memory.*
- *128 Bytes of Scratch pad RAM.*

If you have a need for a fast and dependable video display board, Smoke Signal Broadcasting has what you are looking for. The SMOKE WRITER is right at home when used with a cursor based editor or in a business program that needs protected fields.

SMOKE SIGNAL BROADCASTING

31336 Via Colinas, Westlake Village, CA 91361, (213) 889-9340

'68' Micro Journal
3018 Hamill Rd.
Hixson, TN 37343
USPS 468510

Second Class Postage
Paid at Chattanooga, TN
Pub. No. 468510